Introduction

Motivation

This course focuses on algorithms in signal processing and communications. When training data is available in such systems, we can process the data by first training on historical data and then running a Bayesian scheme, which relies on the statistics being known. A similar Bayesian approach can also be used when the statistics are approximately known.

For example, consider lossy image compression. It is well known that wavelet coefficients of images have a distribution that resembles Laplace,
Equation:

$$f_x\left(x\right)=c_1xe^{-c_2|x|},$$

and the coefficients are approximately independent and identically distributed (i.i.d.). A well-known approach to lossy image compression is to first compute the wavelet coefficients and then compress them using a lossy compressor that is designed for Laplace i.i.d. coefficients [link]. In this approach, the training consists of the body of art that realizes that wavelet coefficients are approximately Laplace i.i.d., and the Bayesan algorithm is a lossy compressor that is designed for this distribution.

However, sometimes the statistics of the input (often called the *source*) are completely unknown, there is no training data, or there is great uncertainty in the statistics. For example, in lossless data compression, we do not know whether a file is an executable, source code, a DNA sequence, contains financial transactions, is text, etc. And even if we know that the file contains text, it has been noted that even different chapters that appear in the same book that is written by the same author may contain different statistics.

For this latter set of problems, the Bayesian approach is useless, because there is no training data. A good alternative approach to Bayesian algorithms is to use *universal algorithms* [link], [link]. These algorithms have good performance irrespective of statistics. In fact, in some cases, these algorithms can achieve (with equality) the theoretically optimum performance in an appropriate asymptotic framework.

In *lossless compression*, universal algorithms have had great impact. For example, the Lempel-Ziv family of algorithms [link], [link] asymptotically achieve the entropy rate [link], [link], which is the best possible compression ratio achievable in lossless compression, despite not knowing the input statistics. Additionally, the Lempel-Ziv algorithms allow efficient implementation.

Overview

The goal of this course is to study universal algorithms, starting from the well-trodden material in lossless compression, and later discussing universal algorithms in other areas of communications and signal processing. Let us overview the material studied during the course. We begin in [link] with a review of some information theory material, including typical sequences and source coding, in order to provide sufficient background. Next, [link] statistical models for data will be described. [link] then presents techniques for universal lossless compression of parametric sources. One approach to universal compression of parametric sources is minimum description length, which compresses the data in order to minimize for the sum of the complexity of the model and the complexity of the data given the parameters of the model. Minimum description length has been used with context tree models to provide universal contextual prediction; context tree approaches are detailed in [link]. We then switch gears in [link] and move beyond lossless compression; universal lossy compression and signal reconstruction are described in detail. Finally, [link] describes Lempel-Ziv algorithms for universal lossless compression based on parsing an input sequence. For convenienve, notation is summarized in [link].

This manuscript is a work in progress, and we expect to expand and improve it during future teachings of the course.

Background

Convergence of random variables

We include some background material for the course. Let us recall some notions of convergence of random variables (RV's).

- A sequence of RV's $\{x_n\}_{n\geq 1}$ converges in probability if $\forall \epsilon \geq 0, \lim_{n\to\infty} \sup \Pr\left(|x_n - \overline{x}\,| > \epsilon\right) = 0$. We denote this by $x_n \overset{P.}{\to} \overline{x}$.
- A sequence of RV's $\{x_n\}_{n\geq 1}$ converges to $\overline{x}$ with probability 1 if $\Pr\left\{x_1, x_2, ... : \lim_{n\to\infty} x_n = \overline{x}\right\} = 1$. We denote this by $x_n \overset{w.p.1}{\longrightarrow} \overline{x}$ or $x_n \overset{a.s.}{\to} \overline{x}$.
- A sequence of RV's $\{x_n\}_{n\geq 1}$ converges to $\overline{x}$ in the ℓ_p sense if $E[|x_n - \overline{x}|^p] \to 0$. We denote this by $x_n \overset{\ell_p}{\to} \overline{x}$.

For example, for $p = 2$ we have mean square convergence, $x_n \overset{m.s.}{\longrightarrow} \overline{x}$. For $p \geq 2$,
Equation:

$$E|x_n, -, \overline{x}|^{p-1} = E\left(|x_n, -, \overline{x}|^p\right)^{\frac{p-1}{p}} \leq \left(E|x_n - \overline{x}|^p\right)^{\frac{p-1}{p}}.$$

Therefore, $x_n \overset{\ell_p}{\to} \overline{x}$ yields $x_n \overset{\ell_{p-1}}{\longrightarrow} \overline{x}$. Note that for convergence in ℓ_1 sense, we have
Equation:

$$\Pr\left(|x_n - \overline{x}\,| > \epsilon\right) \leq \frac{E|x_n - \overline{x}|}{\epsilon} \to 0.$$

Typical Sequences

The following material appears in most textbooks on information theory (c.f., Cover and Thomas [link] and references therein). We include the highlights in order to make these notes self contained, but skip some details and the proofs. Consider a sequence $x = x^n = (x_1, x_2, \ldots, x_n)$, where $x_i \in \alpha$, α is the alphabet, and the cardinality of α is r, i.e., $|\alpha| = r$.

Definition 1 The *type* of x consists of the empirical probabilities of symbols in x,
Equation:

$$P_x(a) = \frac{n_x(a)}{n}, \quad a \in \alpha,$$

where $n_x(a)$ is the *empirical symbol count*, which is the the number of times that $a \in \alpha$ appears in x.

Definition 2 The set of *all possible types* is defined as P_n.

Example:
For an alphabet $\alpha = \{0, 1\}$ we have
$P_n = \left\{\left(\frac{0}{n}, \frac{n}{n}\right), \left(\frac{1}{n}, \frac{n-1}{n}\right), \ldots, \left(\frac{n}{n}, \frac{0}{n}\right)\right\}$. In this case, $|P_n| = n + 1$.

Definition 3 A *type class* T_x contains all $x' \in \alpha^n$, such that P_x=$P_{x'}$,
Equation:

$$T_x = T(P_x) = \left\{x' \in \alpha^n : P_{x'} = P_x\right\}.$$

Example:
Consider $\alpha = 1, 2, 3$ and $x = 11321$. We have $n = 5$ and the empirical counts are $n_x = (3, 1, 1)$. Therefore, the type is $P_x = \left(\frac{3}{5}, \frac{1}{5}, \frac{1}{5}\right)$, and the

type class T_x contains all length-5 sequences with 3 ones, 1 two, and 1 three. That is, $T_x = \{11123, 11132, \ldots, 32111\}$. It is easy to see that $\left|T_x\right| = \frac{5!}{3!1!1!} = 20$.

Theorem 1 The cardinality of the set of all types satisfies $\left|P_n\right| \leq (n+1)^{r-1}$.

The proof is simple, and was given in class. We note in passing that this bound is loose, but it is good enough for our discussion.

Next, consider an i.i.d. source with the following prior,
Equation:

$$Q(x) = \prod_{i=1}^{n} Q(x_i).$$

We note in passing that i.i.d. sources are sometimes called memoryless. Let the entropy be
Equation:

$$H(P_x) = -\Sigma_{a\in\alpha} \frac{n_x(a)}{n} \log\left(\frac{n_x(a)}{n}\right),$$

where we use base-two logarithms throughout. We are studying the entropy $H(P_x)$ in order to show that it is the fundamental performance limit in lossless compression. Σ find me

We also define the divergence as
Equation:

$$D(P_x \parallel Q_x) = \Sigma_{a\in\alpha} P_x \log\left(\frac{P_x}{Q_x}\right).$$

It is well known that the divergence is non-negative,
Equation:

$$D(P_x \parallel Q_x) \geq 0.$$

Moreover, $D(P \parallel Q) = 0$ only if the distributions are identical.

Claim 1 The following relation holds,
Equation:

$$Q\left(x\right) = 2^{-n[H(P_x)+D(P_x\|Q(x))]}.$$

The derivation is straightforward,
Equation:

$$\begin{array}{rcl} Q(x) & = & \Pi_{a\in\alpha}Q(a)^{n_x(a)} \\ & = & 2^{\Sigma_{a\in\alpha}n_x(a)\text{log}Q(a)} \\ & = & 2^{n\Sigma P_x(a)\left(\text{log}\frac{Q}{P}+\text{log}P\right)} \\ & = & 2^{-n[H(P_x)+D(P_x\|Q(x))]}. \end{array}$$

Seeing that the divergence is non-negative [link], and zero only if the distributions are equal, we have $Q\left(x\right) \leq P_x\left(x\right)$. When $P_x = Q$ the divergence between them is zero, and we have that $P_x\left(x\right) = Q_x = 2^{-nH(P_x)}$.

The proof of the following theorem was discussed in class.

Theorem 2 The cardinality of the type class $T(P_x)$ obeys,
Equation:

$$\left(n+1\right)^{-(r-1)} \cdot 2^{nH(P_x)} \leq \left|T\left(P_x\right)\right| \leq 2^{nH(P_x)}.$$

Having computed the probability of x and cardinality of its type class, we can easily compute the probability of the type class.

Claim 2 The probability $Q(T\left(P_x\right)$ of the type class $T(P_x)$ obeys,
Equation:

$$\left(n+1\right)^{-\left(r-1\right)}\cdot 2^{-nD\left(P_x\|Q_x\right)}\le Q\left(T\left(P_x\right)\right)\le 2^{-nD\left(P_x\|Q_x\right)}.$$

Consider now an event A that is a union over $T(P_x)$. Suppose $T(Q)\not\subseteq A$, then A is rare with respect to (w.r.t) the prior Q. and we have $\lim_{n\to\infty}Q\left(A\right)=0$. That is, the probability is concentrated around Q. In general, the probability assigned by the prior Q to an event A satisfies
Equation:

$$\begin{aligned} Q(A) &= \Sigma_{x\in A}Q\left(x\right)=\Sigma_{T\left(P_x\right)\subseteq A}Q\left(T\left(P_x\right)\right)\\ &\doteq \Sigma_{T\left(P_x\right)\subseteq A}2^{-nD\left(P_x\|Q\right)}\\ &\doteq 2^{-n\cdot\min_{p\in A}D\left(P\|Q\right)}, \end{aligned}$$

where we denote $a_n\doteq b_n$ when $\frac{1}{n}\log\left(\frac{a_n}{b_n}\right)\to 0$.

Fixed and Variable Length Coding

Fixed to fixed length source coding: As before, we have a sequence x of length n, and each element of x is from the alphabet α. A *source code* maps the input $x^n\in r^n$ to a set of 2^{Rn} bit vectors, each of length Rn. The rate R quantifies the number of output bits of the code per input element of x. [footnote] That is, the output of the code consists of nR bits. If n and R is fixed, then we call this a *fixed to fixed* length source code.
We assume without loss of generality that $Rn\in\mathbb{Z}$. If not, then we can round Rn up to $\lceil Rn\rceil$, where $\lceil\cdot\rceil$ denotes rounding up.

The decoder processes the nR bits and yields $\widehat{x}\in\alpha^n$. Ideally we have that $\widehat{x}=x$, but if $2^{nR}<r^n$ then there are inputs that are not mapped to any

output, and $\widehat{x}$ may differ from x. Therefore, we want $\Pr\left(\widehat{x} \neq x\right)$ to be small. If R is too small, then the error probability will go to 1. On the other hand, sufficiently large R will drive this error probability to 0 as n is increased.

If $\log\left(r\right) > R$ and $\Pr\left(\widehat{x} \neq x\right)$ is vanishing as n is increased, then we are compressing, because $2^{\log(r)n} = r^n > 2^{Rn}$, where r^n is the number of possible inputs x and there are 2^{Rn} possible outputs.

What is a good fixed to fixed length source code? One option is to map $2^{Rn} - 1$ outputs to inputs with high probabilities, and the last output can be mapped to a “don't care" input. We will discuss the performance of this style of code.

An input $x \in r^n$ is called δ-typical if $Q\left(x\right) > 2^{-(H+\delta)n}$. We denote the set of δ-typical inputs by $T_Q\left(\delta\right)$, this set includes the type classes whose empirical probabilities are equal (or closest) to the true prior $Q(x)$. Note that for each type class T_x, all inputs $x' \in T_x$ in the type class have the same probability, i.e., $Q\left(x'\right) = Q\left(x\right)$. Therefore, the set $T_Q\left(\delta\right)$ is a union of type classes, and can be thought of as an event A ([link]) that contains type classes consisting of high-probability sequences. It is easily seen that the event A contains the true i.i.d. distribution Q, because sequences whose empirical probabilities satisfy $P_x = Q$ also satisfy
Equation:

$$Q\left(x\right) = 2^{-Hn} > 2^{-(H+\delta)n}.$$

Using the principles discussed in [link], it is readily seen that the probability under the prior Q of the inputs in $T_Q\left(\delta\right)$ satisfies $Q(T_p\left(\delta\right)) = Q\left(A\right) \to 1$ when $n \to \infty$. Therefore, a code $\mathscr{C}$ that enumerates $T_Q\left(\delta\right)$ will encode x correctly with high probability.

The key question is the size of $\mathscr{C}$, or the cardinality of $T_Q\left(\delta\right)$. Because each $x \in T_Q\left(\delta\right)$ satisfies $Q\left(x\right) > 2^{(-H+\delta)n}$, and $\sum_{x \in T_Q(\delta)} Q\left(x\right) \leq 1$, we have $\left|T_Q\left(\delta\right)\right| < 2^{(H+\delta)n}$. Therefore, a rate $R \geq H + \delta$ allows *near-*

lossless coding, because the probability of error vanishes (recall that $Q\left(\left(T_p\left(\delta\right)\right)^C\right) \to 0$, where $\left(\cdot\right)^C$ denotes the complement).

On the other hand, a rate $R \leq H - \delta$ will not allow lossless coding, and the probability of error will go to 1. We will see this intuitively. Because the type class whose empirical probability is Q dominates, a type class T_x whose sequences have larger probability, e.g., $Q\left(x\right) > 2^{-\left(H-\delta\right)n}$, will have small probability in aggregate. That is,
Equation:

$$\sum_{x:Q(x)>2^{-n(H-\delta)}} Q\left(x\right) \xrightarrow{n\to\infty} 0.$$

In words, choosing a code $\mathscr{C}$ with rate $R = H - \delta$ that contains the words x with highest probability will fail, it will not cover enough probabilistic mass. We conclude that near-lossless coding is possible at rates above H and impossible below H.

To see things from a more intuitive angle, consider the definition of entropy, $H\left(Q\right) = -\sum_{a\in\alpha} Q\left(a\right)\log\left(Q\left(a\right)\right)$. If we consider each bit as reducing uncertainty by a factor of 2, then the average log-likelihood of a length-n input x generated by Q satisfies
Equation:

$$\begin{aligned} E[-\log\left(\Pr\left(x\right)\right)] &= E\left[-\log\left(\prod_{i=1}^{n} Pr\left(x_i\right)\right)\right] \\ &= -\sum_{i=1}^{n} E\left[\log\left(Q\left(x_i\right)\right)\right] \\ &= -\sum_{i=1}^{n}\sum_{a\in\alpha} Q\left(a\right)\cdot\log\left(Q\left(a\right)\right) \\ &= nH. \end{aligned}$$

Because the expected log-likelihood of x is nH, it will take nH bits to reduce the uncertainty by this factor.

Fixed to variable length source coding: The near-lossless coding above relies on enumerating a collection of high-probability codewords $T_Q\left(\delta\right)$. However, this approach suffers from a troubling failure for $x \notin T_Q\left(\delta\right)$. To solve this problem, we incorporate a code that maps x to an output consisting of a *variable* number of bits. That is, the length of the code will be approximately nH on average, but could be greater or lesser.

One possible variable length code is due to Shannon. Consider all possible $x \in \alpha^n$. For each x, allocate $\lceil -\log\left(Q(x)\right)\rceil$ bits to x. It can be shown that it is possible to construct an invertible (uniquely decodable) code as long as the length of the code $l(x)$ in bits allocated to each x satisfies
Equation:

$$\sum_x 2^{-l(x)} \leq 1.$$

This result is known as the Kraft Inequality. Seeing that
Equation:

$$\begin{aligned}\sum_x 2^{-l(x)} &= \sum_x 2^{-\lceil -\log(Q(x))\rceil} \\ &\leq \sum_x 2^{-(-\log(Q(x)))} \\ &= \sum_x Q\left(x\right) = 1,\end{aligned}$$

we see that the length allocation we suggested satisfies the Kraft Inequality. Therefore, it is possible to construct an invertible (and hence lossless) code with lengths upper bounded by
Equation:

$$l_x = \lceil -\log\left(Q\left(x\right)\right)\rceil \leq -\log\left(Q\left(x\right)\right) + 1,$$

and we have
Equation:

$$E[l(x)] \leq E[-\log(Q(x))] + 1 = nH + 1.$$

This simple construction approaches the entropy up to 1 bit.

Unfortunately, a Shannon code is impractical, because it requires to construct a code book of exponential size $|\alpha|^n$. Instead, arithmetic codes [link] are used; we discussed arithmetic codes in detail in class, but they appear in all standard text books and so we do not describe them here.

Source models

For i.i.d. sources, $D(P_1(x^n)||P_2(x^n)) = nD(P_1(x_i)||P_2(x_i))$, which means that the divergence increases linearly with n. Not only does the divergence increase, but it does so by a constant per symbol. Therefore, based on typical sequence concepts that we have seen, for an x^n generated by P_1, its probability under P_2 vanishes. However, we can construct a distribution Q whose divergence with both P_1 abd P_2 is small,
Equation:

$$Q(x^n) = \frac{1}{2}P_1(x^n) + \frac{1}{2}P_2(x^n).$$

We now have for P_1,
Equation:

$$\begin{aligned}\frac{1}{n}D(P_1^n||Q) &= \frac{1}{n}E\left[\log\frac{P_1(x^n)}{\frac{1}{2}P_1(x^n)+\frac{1}{2}P_2(x^n)}\right]\\ &\le \frac{1}{n}\log(2) = \frac{1}{n}.\end{aligned}$$

On the other hand, $\frac{1}{n}D(P_1(x_1^n)||Q(x_1^n)) \ge 0$ [link], and so
Equation:

$$\frac{1}{n} \ge \frac{1}{n}D(P_1(x_1^n)||Q(x_1^n)) \ge 0.$$

By symmetry, we see that Q is also close to P_2 in the divergence sense.

Intuitively, it might seem peculiar that Q is close to both P_1 and P_2 but they are far away from each other (in divergence terms). This intuition stems from the triangle inequality, which holds for all metrics. The contradiction is resolved by realizing that the divergence is not a metric, and it does not satisfy the triangle inequality.

Note also that for two i.i.d. distributions P_1 and P_2, the divergence
Equation:

$$D(P_1(x^n)||P_2(x^n)) = nD(P_1||P_2)$$

is linear in n. If Q were i.i.d., then $D(P_1(x^n)\parallel Q(x_1^n))$ must also be linear in n. But the divergence is not increasing linearly in n, it is upper bounded by 1. Therefore, we conclude that $Q(\cdot)$ is not an i.i.d. distribution. Instead, Q is a distribution that contains memory, and there is dependence in Q between collections of different symbols of x in the sense that they are either all drawn from P_1 or all drawn from P_2. To take this one step further, consider K sources with
Equation:

$$Q\left(x^{n}\right)=\sum_{i=1}^{K}\frac{1}{K}P_{i}\left(x^{n}\right),$$

then in an analogous manner to before it can be shown that
Equation:

$$D(P_{i}\left(x_{1}^{n}\right)||Q\left(x_{1}^{n}\right))\le\frac{1}{n}\log\left(K\right).$$

Sources with memory: Instead of the memoryless (i.i.d.) source,
Equation:

$$P\left(x^{n}\right)=\prod_{i=1}^{n}P\left(x_{i}\right),$$

let us now put forward a statistical model with memory,
Equation:

$$P\left(x^{n}\right)=\prod_{i=1}^{n}P\left(x_{i}\middle|x_{1}^{i-1}\right).$$

Stationary source: To understand the notion of a stationary source, consider an infinite stream of symbols, $\ldots,x_{-1},x_{0},x_{1},\ldots$. A complete probabilistic description of a stationary distribution is given by the collection of all marginal distribution of the following form for all t and n,
Equation:

$$P_{X_{t},X_{t+1},\ldots,X_{t+n-1}}\left(x_{t},x_{t+1},\ldots,x_{t+n-1}\right).$$

For a stationary source, this distribution is independent of t.

Entropy rate: We have defined the first order entropy of an i.i.d. random variable [link], and let us discuss more advanced concepts for sources with memory. Such definitions appear in many standard textbooks, for example that by Gallager [link].

1. The *order-n entropy* is defined,
   Equation:

   $$H_{n}=\frac{1}{n}H\left(x_{1},\ldots,x_{n}\right)=-\frac{1}{n}E\left[\log\left(P\left(x_{1},\ldots,x_{n}\right)\right)\right].$$

2. The *entropy rate* is the limit of order-n entropy, $\overline{H}$ $=\lim_{n\to\infty}$ H_{n}. The existence of this limit will be shown soon.

3. *Conditional entropy* is defined similarly to entropy as the expectation of the log of the conditional probability,
   Equation:

$$H\left(x_n|x_1,\ldots,x_{n-1}\right) = -\frac{1}{n}E\left[\log\left(P\left(x_n|x_1,\ldots,x_{n-1}\right)\right)\right],$$

   where expectation is taken over the joint probability space, $P(x_1,\ldots,x_n)$.

The entropy rate also satisfies $\overline{H} = \lim_{n\to\infty} H\left(x_n|x_1,\ldots,x_n\right)$.

Theorem 3 For a stationary source with bounded first order entropy, $H_1\left(x\right) < \infty$, the following hold.

1. The conditional entropy $H(x_n|x_1,\ldots,x_{n-1})$ is monotone non-increasing in n.
2. The order-n entropy is not smaller than the conditional entropy,
   Equation:

$$H_n\left(x\right) \geq H\left(x_n|x_1,\ldots,x_{n-1}\right).$$

3. The order-n entropy $H_n\left(x\right)$ is monotone non-increasing.
4. $\overline{H}\left(x\right) = \lim_{n\to\infty} H_n\left(x\right) = \lim_{n\to\infty} H\left(x_n|x_1,\ldots,x_{n-1}\right)$.

Proof. **Part (1):**
Equation:

$$\begin{aligned} H(x_n|x_1,\ldots,x_{n-1}) &\leq H(x_n|x_2,\ldots,x_{n-1}) \\ &= H(x_{n-1}|x_1,\ldots,x_{n-2}) \\ &\leq \ldots \leq H\left(x_2|x_1\right) \leq H\left(x_1\right). \end{aligned}$$

Part (2):
Equation:

$$\begin{aligned} H_n\left(x\right) &= \frac{1}{n}\left[H\left(x_1\right) + H\left(x_2|x_1\right) + \ldots + H\left(x_n|x_1,\ldots,x_{n-1}\right)\right] \\ &\geq \frac{1}{n}\left[H\left(x_n|x_1,\ldots,x_{n-1}\right) + \ldots + H\left(x_n|x_1,\ldots,x_{n-1}\right)\right] \\ &= H(x_n|x_1,\ldots,x_{n-1}). \end{aligned}$$

Part (3): This comes from the fist equality in the proof of (2), because we have the average of a monotonely non-increasing sequence.

Part (4): Both sequences are monotone non-increasing (parts (1) and (3)) and bounded below (by zero). Therefore, they both have a limit. Denote $\overline{H}\left(x\right) = \lim_{n\to\infty} H_n\left(x\right)$ and $\widetilde{H}\left(x\right) = \lim_{n\to\infty} H\left(x_n|x_1,\ldots,x_{n-1}\right)$.

Owing to part(2), $\overline{H} \geq \widetilde{H}$. Therefore, it suffices to prove $\widetilde{H} \geq \overline{H}$.
Equation:

$$\begin{aligned} H_{n+m}(x) &= \frac{1}{n+m}\left[H\left(x_1^{n-1}\right) + \sum_{i=n}^{n+m} H\left(x_i|x_1,\ldots,x_{i-1}\right)\right] \\ &\leq \frac{H\left(x_1^{n-1}\right)}{n+m} + \frac{m+1}{n+m} H\left(x_i|x_1,\ldots,x_{i-1}\right). \end{aligned}$$

Now fix n and take the limit for large m. The inequality $\overline{H} \leq \widetilde{H}$ appears, which proves that both limits are equal.

Coding theorem: Theorem 3 yields for fixed to variable length coding that for a stationary source, there exists a lossless code such that the compression rate ρ_n obeys,
Equation:

$$\rho_n = \frac{E[l\left(x_1,\ldots,x_n\right)]}{n} \leq H_n(x) + \frac{1}{n}.$$

This can be proved, for example, by choosing $l\left(x_1,\ldots,x_n\right) = \lceil -\log P\left(x_1,\ldots,x_n\right)\rceil$, which is a Shannon code. As n is increased, the compression rate ρ_n converges to the entropy rate.

We also have a converse theorem for lossless coding of stationary sources. That is, $\rho_n \geq H_n(x) \geq \overline{H}$.

Stationary Ergodic Sources

Consider the sequence $x = (\ldots, x_{-1}, x_0, x_1, \ldots)$. Let $x' = S_x$ denote a step $\forall n \in \mathbb{Z}$, $x'_n = x_{n+1}$, where S_x^i takes i steps. Let $f_k(x)$ be a function that operates on coordinates $(x_0, \ldots, x_{k-1})$. An ergodic source has the property that empirical averages converge to statistical averages,
Equation:

$$\frac{1}{n}\sum_{i=0}^{n-1} f_k\left(S_x^i\right) \xrightarrow{a.s., n\rightarrow\infty} Ef_k(x).$$

In block codes we want
Equation:

$$\frac{1}{nN}\sum_{i=0}^{n-1} l\left(x_{iN+1},\ldots,x_{(i+1)N}\right) \overset{\text{a.s.}}{=} H.$$

We will be content with convergence in probability, and a.s. convergence is better.

Theorem 4 Let X be a stationary ergodic source with $H_1(x) < \infty$, then for every $\epsilon > 0, \delta > 0$, there exists $n_0(\delta, \epsilon)$ such that $\forall n \geq n_0(\delta, \epsilon)$,
Equation:

$$\Pr\left\{\left|\frac{1}{n} I(x_1, \ldots, x_n) - \overline{H}\right|\right\} \leq \epsilon,$$

where $I(x_1, \ldots, x_n) = -\log(\Pr(x_1, \ldots, x_n))$.

The proof of this result is quite lengthy. We discussed it in detail, but skip it here.

Theorem 4 is called the ergodic theorem of information theory or the ergodic theorem of entropy. Shannon (48') proved convergence in probability for stationary ergodic Markov sources. McMillan (53') proved L^1 convergence for stationary ergodic sources. Brieman (57'/60') proved convergence with probability 1 for stationary ergodic sources.

Parametric Models of Information Sources

In this section, we will discuss several parametric models and see what their entropy rate is.

Memoryless sources: We have seen for memoryless sources,
Equation:

$$p(x) = \prod_{i=1}^{n} p(x_i),$$

where there are $r - 1$ parameters in total,
Equation:

$$\theta = \{p(a), a = 1, 2, \ldots, r-1\},$$

the parameters are denoted by θ, and $\alpha = \{1, 2, \ldots, r\}$ is the alphabet.

Markov sources: The distribution of a Markov source is defined as
Equation:

$$p(x_1, x_2, \ldots, x_n) = p(x_1, \ldots, x_k) \prod_{i=k+1}^{n} p\left(x_i \middle| x_{i-k}^{i-1}\right),$$

where $n \geq k$. We must define $\{p(a_1, a_2, \ldots, a_k)\}_{(a_1, a_2, \ldots a_k) \in \alpha^k}$ initial probabilities and transition probabilities, $\left\{p\left(a_{k+1} \middle| a_1^k\right)\right\}$. There are $r^k - 1$ initial probabilities and $(r-1)r^k$

transition probabilities, giving a total of $r^{k+1} - 1$ parameters. Note that
Equation:

$$E\left\{-\log p\left(x_i | x_{i-k}^{i-1}\right)\right\} = H\left(X_i | X_{i-k}^{i-1}\right) \xrightarrow{k\to\infty} \overline{H}\left(X\right).$$

Therefore, the space of Markov sources covers the stationary ergodic sources in the limit of large k.

Unifilar sources: For unifilar sources, it is possible to reconstruct the set of states that a source went through by looking at the output sequence. In the Markov case we have $S_i = \left(X_{i-k}^{i-1}\right)$, but in general it may be more complicated to determine the state.

To put us on a concrete basis for analysis of unifilar sources, consider a source with M states, $S = \{1, 2, \ldots, M\}$, and an alphabet $\alpha = \{1, 2, \ldots, r\}$. In each time step, the source outputs a symbol and moves to a new state. Denote the output sequence by $x = x_1 x_2 \cdots x_n$, and the state sequence by $s = s_1 s_2 \cdots s_n$, where $s_i \in S$ and $x_i \in \alpha$. Denote also
Equation:

$$\begin{aligned} q\big(s|s'\big) &= \mathrm{Pr}\big\{S_t = s|S_{t-1} = s'\big\} \\ &= \mathrm{Pr}\big\{S_t = s|S_{t-1} = s', S_{t-2}, \cdots\big\}. \end{aligned}$$

This is a first-order time-homogeneous Markov source. The probability that the next symbol is a follows,
Equation:

$$\begin{aligned} p(a|s) &= \mathrm{Pr}\{X_t = a|S_t = s\} \\ &= \mathrm{Pr}\{X_t = a|S_t = s, X_{t-1}, S_{t-1}, \cdots\}. \end{aligned}$$

There exists a deterministic function,
Equation:

$$S_t = g\left(S_{t-1}, X_{t-1}\right),$$

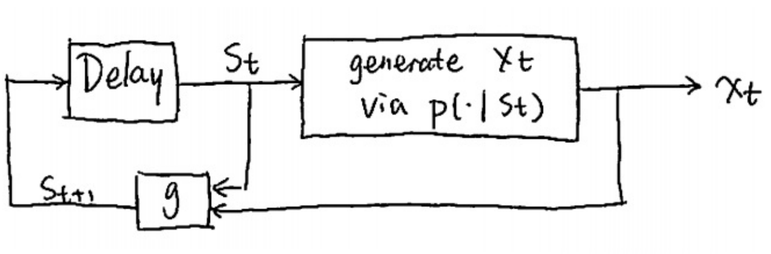
this is called the next state function. Given that we start at some state $S_1 = s_1$, the probability for the sequence of states s_1, ..., s_n is given by
Equation:

$$p\left(X_1^n | S_1\right) = \prod_{t=1}^{n} p\left(X_t | S_t\right).$$

Note the relation
Equation:

$$q\left(s|s'\right)=\sum_{a:g(s',a)=s}p\left(a|s'\right).$$

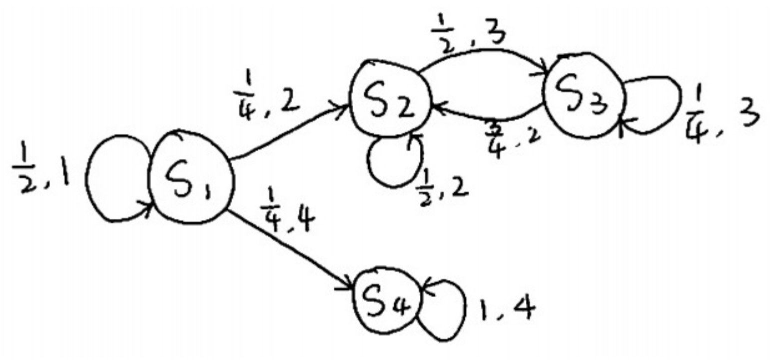
To summarize, unifilar sources can be described by a state machine style of diagram as illustrated in [link].



State machine for selecting the state of a unifilar source.

Given that an initial state was fixed, a unifilar source with M states and an alphabet of size r can be expressed with $M(r-1)$ parameters. If the initial state is a random variable, then there are $M-1$ parameters that define probabilities for the initial state, giving $M(r-1)+M-1=Mr-1$ parameters in total. In the Markov case, we have $M=r^k$, it is a special type of unifilar source.

Example:
For the unifilar source that appears in [link], the states can be discerned from the output sequence. Let us follow up on this example while discussing more properties of unifilar sources.

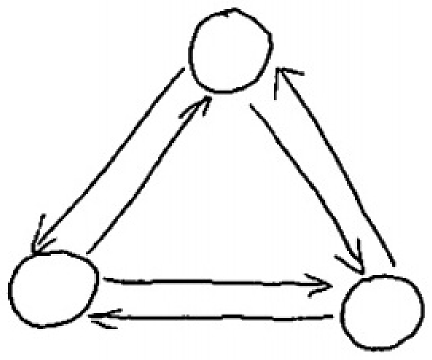


Unifilar source for [link].

A state S is called *transient* if once we leave it we cannot go back; S_1 is such a state in the exaple. A subset E of states is called *irreducible* if from every state in E it is possible to move to every other state in E in a finite number of steps, and it is impossible to leave E. We see that $\{S_2, S_3\}$ and $\{S_4\}$ are irreducible sets. The states can be decomposed (in a single unique way) into a set of transient states and irreducible components. It can be shown that with probability 1 the time spent in transient states is finite, and after that the source enters one of the irreducible components and stays there forever. Finally, in an ergodic source there may only be one irreducible component. Therefre, our example source is not ergodic because it contains two such components.

Given that we are in some state s, which belongs to an irreducible component, the number of time steps until we return to s for the first time is a random variable called *recurrence time*. The *period* of an irreducible component is the largest integer that divides all possible recurrence times.

Example:
In the unifilar source in [link] the period is 1, because recurrence time could be 2 or 3. However, it is impossible for the recurrence time to be 1.



Unifilar source for [link].

An irreducible component with period at least 2 is called periodic. In contrast, a component with period 1 is called aperiodic or ergodic, because the probabilities in being in each of the states eventually converge to the statistical averages. Note that a periodic component cannot yield a stationary distribution.

An irreducible component E contains asymptotic state probabilities given by the solution of the equations,
Equation:

$$\sum_{s'\in E} q\left(s'\right) q\left(s|s'\right) = q\left(s\right), \quad \forall s \in E.$$

Additionally,
Equation:

$$\lim_{t\to\infty} \Pr\left\{S_t = s|S_1 = s'\right\} = q_\infty\left(s\right),$$

for aperiodic E. The convergence of $\Pr\{S_t = s|S_1 = s'\}$ to $q(s)$ is exponential in t, and depends on the eigen-values of the transition probability matrix. In periodic components we do not have stationarity, but the following weaker property still holds,
Equation:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{t} \Pr\left\{S_i = s\right\} = q_\infty\left(s\right).$$

The entropy rate of the output of a unifilar source at state s is given by,
Equation:

$$H\left(X|S = s\right) = -\sum_{a\in A} p\left(a|s\right)\cdot \log p\left(a|s\right).$$

To see this, we begin with a lemma.

Lemma 1 For a unifilar source,
Equation:

$$H\left(X_n\middle|X_1^{n-1}, S_1 = s'\right) = \sum_{s\in S} \Pr\left\{S_n = s|S_1 = s'\right\} H\left(X|s\right).$$

Proof $\Pr\left\{X_n\middle|X_1^{n-1}, S_1\right\} = \Pr\left\{X_n\middle|S_n, X_1^{n-1}, S_1\right\}$, because S_n is deterministic in S_{n-1}, X_{n-1}. Therefore, owing to the Markov property,
Equation:

$$\Pr\left(X_n\middle|X_1^{n-1}, S_1\right) = \Pr\left(X_n|S_n\right).$$

We can now compute the entropy,

Equation:

$$\begin{aligned} H\left(X_n \middle| X_1^{n-1}, S_1 = s_1\right) &= -\sum_{(X_1^n, S_n)} \Pr\left(X_1^n, S_n | S_1\right) \log\left(\Pr\left(X_n \middle| X_1^{n-1}, S_1\right)\right) \\ &= -\sum_{(X_1^n, S_n)} \Pr\left(X_1^{n-1}, S_n \middle| S_1\right) \sum_{X_n} \Pr\left(X_n | S_n\right) \log\left(\Pr\left(X_n | S_n\right)\right) \\ &= \sum_{S} \Pr\left(S_n = s | S_1\right) \cdot H\left(X | S_n = s\right). \end{aligned}$$

Having proved the lemma, we can move forward with the computation of entropy rate. Note that

Equation:

$$\begin{aligned} \frac{1}{n} H\left(X_1, \ldots, X_n | S_1\right) &= \frac{1}{n} \sum_{l=1}^{n} H\left(X_l \middle| X_1^{l-1}, S_1\right) \\ &= \frac{1}{n} \sum_{l=1}^{n} \sum_{s \in S} \Pr\left(S_l = s\right) \cdot H\left(X | s\right) \\ &= \sum_{s \in S} \left[\frac{1}{n} \sum_{l=1}^{n} \Pr\left(S_l = s\right)\right] H\left(X | S = s\right), \end{aligned}$$

and

Equation:

$$\frac{1}{n} \sum_{l=1}^{n} \Pr\left(S_l = s\right) \to q_\infty\left(s\right).$$

Therefore,

Equation:

$$\lim_{n \to \infty} \frac{1}{n} H\left(X_1^n | S_1\right) = \sum_{s \in S} q_\infty\left(s\right) H\left(X | s\right).$$

It is not complicated to show that this is the entropy rate of the unifilar source X, provided that it is irreducible and aperiodic.

Tree Sources: Tree sources are very useful in modeling and processing text and data files in general. Consequently, tree sources have been studied in great detail. We will introduce these source models here and later in [link] discuss algorithms for universal lossless compression of these sources.

In a tree source, states are arranged as *contexts*, where a context consists of the last several symbols generated by the source. The unique property of a context tree source is that states are arranged as leaves of a tree, and the state can be identified by following labels of the branches of the tree based on previous symbols, until we reach a leaf.

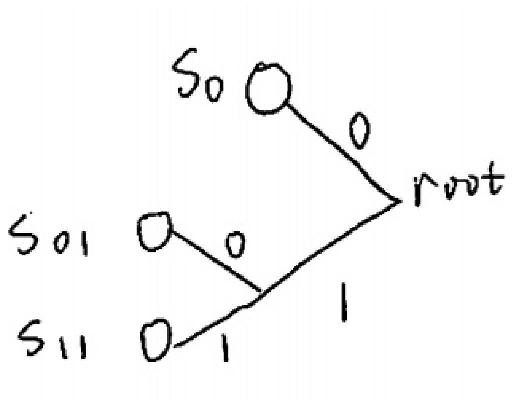
Example:
[link] shows a simple tree source. For this source, a typical input might be

Equation:

$$x = 011001010.$$

This input sequence corresponds to the following sequence of states,
Equation:

$$s = s_0 s_{01} s_{11} s_0 s_0 s_{01} s_0 s_{01} s_0.$$



Tree source for [link].

As before, for every state (leaf) there exist conditional probabilities for generating the next symbol. That said, and maybe surprisingly, a tree source is not necessarily a unifilar source. On the one hand, states can be observed from the source symbols. On the other hand, when transitioning from a “shallow" leaf to a deep leaf in the tree, the symbol currently being generated does not contain enough information to define the new state. To address this problem, extra states can be added to the tree. On the other hand, adding extra states to a tree source might make the context tree less efficient, because it increases the number of states; see for example the detailed discussion of expansion of tree machines by Martín et al. [link].

Hidden Markov models: Consider the following state transition probabilities,
Equation:

$$\begin{aligned}\Pr\big(X_t = a, S_t = s \big| S_{t-1} = s', S_{t-2}, X_{t-1}, \dots\big) &= \Pr\big(X_t = a, S_t = s \big| S_{t-1} = s'\big) \\ &= \Pr\left(a, s \middle| s'\right).\end{aligned}$$

The parameter space can be defined,
Equation:

$$\theta = \left\{\Pr\left(a, s \middle| s'\right)\right\},$$

there are $M(M \cdot r - 1)$ parameters in total.

Note that
Equation:

$$\Pr\left(X_1^n, S_1^n | S_0\right) = \prod_{t=1}^{n} \Pr\left(X_t, S_t | S_{t-1}\right)$$

and
Equation:

$$\begin{aligned}\Pr\left(X_1^n | S_0\right) &= \sum_{s_1^n} \Pr\left(X_1^n, S_1^n | S_0\right) \\ &= \sum_{s_1^n} \prod_{t=1}^{n} \Pr\left(X_t, S_t | S_{t-1}\right).\end{aligned}$$

It is important to emphasize that the next state is not deterministic. This type of behavior is called a hidden Markov model (HMM).

What are the typical sets for HMM sources? Recall that for an i.i.d. source we saw
Equation:

$$\Pr\left(X_1^n\right) = \prod_{a \in A} \Pr\left(a\right)^{n_x(a)},$$

this is the main concept underlying the method of types. For a unifilar source,
Equation:

$$\begin{aligned}\Pr\left(X_1^n\right) &= \prod_{t=1}^{n} \Pr\left(X_t|S_t\right) \\ &= \prod_{A\times S} \Pr\left(a|s\right)^{n_X(a|s)}.\end{aligned}$$

For an HMM, there is no such partition into typical sets.

Universal coding for classes of sources

We have discussed several parametric sources, and will now start developing mathematical tools in order to investigate properties of universal codes that offer universal compression w.r.t. a class of parametric sources.

Preliminaries

Consider a class Λ of parametric models, where the parameter set θ characterizes the distribution for a specific source within this class, $\{p_\theta\left(\cdot\right), \theta \in \Lambda\}$.

Example:
Consider the class of memoryless sources over an alphabet $\alpha = \{1, 2, \ldots, r\}$. Here we have

Equation:

$$\theta = \{p(1), p(2), \ldots, p(r-1)\}.$$

The goal is to find a fixed to variable length lossless code that is independent of θ, which is unknown, yet achieves
Equation:

$$E_\theta \frac{l(X_1^n)}{n} \xrightarrow{n\to\infty} \overline{H_\theta}\left(X\right),$$

where expectation is taken w.r.t. the distribution implied by θ. We have seen for
Equation:

$$p\left(x\right) = \frac{1}{2} p_1\left(x\right) + \frac{1}{2} p_2\left(x\right)$$

that a code that is good for two sources (distributions) p_1 and p_2 exists, modulo the one bit loss [link]. As an expansion beyond this idea, consider
Equation:

$$p\left(x\right)=\int_{\Lambda}dw\left(\theta\right)p_{\theta}\left(X\right),$$

where $w(\theta)$ is a prior.

Example:
Let us revisit the memoryless source, choose $r=2$, and define the scalar parameter

Equation:

$$\theta=\Pr\left(X_i=1\right)=1-\Pr\left(X_i=0\right).$$

Then
Equation:

$$p_{\theta}\left(x\right)=\theta^{n_X(1)}\cdot\left(1-\theta\right)^{n_X(0)}$$

and
Equation:

$$p\left(x\right)=\int_0^1 d\theta\cdot\theta^{n_X(1)}\cdot\left(1-\theta\right)^{n_X(0)}.$$

Moreover, it can be shown that
Equation:

$$p\left(x\right)=\frac{n_X\left(0\right)!n_X\left(1\right)!}{\left(n+1\right)!},$$

this result appears in Krichevsky and Trofimov [link].

Is the source X implied by the distribution $p(x)$ an ergodic source? Consider the event $\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}X_i\le\frac{1}{2}$. Owing to symmetry, in the limit of large n the probability of this event under $p(x)$ must be $\frac{1}{2}$,
Equation:

$$\Pr\left\{\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}X_i\le\frac{1}{2}\right\}=\frac{1}{2}.$$

On the other hand, recall that an ergodic source must allocate probability 0 or 1 to this flavor of event. Therefore, the source implied by $p(x)$ is not ergodic.

Recall the definitions of $p_\theta\left(x\right)$ and $p(x)$ in [link] and [link], respectively. Based on these definitions, consider the following,
Equation:

$$\begin{aligned}H_\theta\left(X_1^n\right)&=-\sum_{X_1^n\in A^n}p_\theta\left(X_1^n\right)\log p_\theta\left(X_1^n\right)=H\left(X_1^n|\Theta=\theta\right),\\H(X_1^n)&=-\sum_{X_1^n}p\left(X_1^n\right)\log p\left(X_1^n\right),\\H(X_1^n|\Theta)&=\int_\Lambda dw\left(\theta\right)\cdot H\left(X_1^n|\Theta=\theta\right).\end{aligned}$$

We get the following quantity for mutual information between the random variable Θ and random sequence X_1^N,
Equation:

$$I\left(\Theta;X_1^n\right)=H\left(X_1^n\right)-H\left(X_1^n|\Theta\right).$$

Note that this quantity represents the gain in bits that the parameter θ creates; more about this quantity will be mentioned later.

Redundancy

We now define the conditional *redundancy*,
Equation:

$$r_n\left(l,\theta\right)=\frac{1}{n}[E_\theta\left(l\left(X_1^n\right)\right)-H_\theta\left(X_1^n\right)],$$

this quantifies how far a coding length function l is from the entropy where the parameter θ is known. Note that
Equation:

$$l\left(X_1^n\right)=\int_\Lambda dw\left(\theta\right)E_\theta\left(l\left(X_1^n\right)\right)\geq H\left(X_1^n|\theta\right).$$

Denote by c_n the collection of lossless codes for length-n inputs, and define the expected redundancy of a code $l\in C_n$ by
Equation:

$$\begin{array}{rcl}R_n^-\left(w,l\right) & = & \int_\Lambda dw\left(\theta\right)r_n\left(l,\theta\right),\\ R_n^-\left(w\right) & = & \inf\limits_{l\in C_n}R_n^-\left(w,l\right).\end{array}$$

The asymptotic expected redundancy follows,
Equation:

$$R^-\left(w\right)=\lim_{n\to\infty}R_n^-\left(w\right),$$

assuming that the limit exists.

We can also define the minimum redundancy that incorporates the worst prior for parameter,
Equation:

$$R_n^- = \sup_{w \in W} R_n^- (w),$$

while keeping the best code. Similarly,
Equation:

$$R^- = \lim_{n \to \infty} R_n^- .$$

Let us derive R_n^-,
Equation:

$$\begin{aligned} R_n^- &= \sup_w \inf_l \int_\Lambda dw(\theta) \frac{1}{n} [E_\theta (l(X_1^n)) - H(X_1^n | \Theta = \theta)] \\ &= \sup_w \inf_l \frac{1}{n} E_p [l(X_1^n) - H(X_1^n | \Theta)] \\ &= \sup_w \frac{1}{n} [H(X_1^n) - H(X_1^n | \Theta)] \\ &= \sup_w \frac{1}{n} I(\Theta; X_1^n) = \frac{C_n}{n}, \end{aligned}$$

where C_n is the capacity of a channel from the sequence x to the parameter [link]. That is, we try to estimate the parameter from the noisy channel.

In an analogous manner, we define
Equation:

$$\begin{aligned}R_n^+ &= \inf_l \sup_{\theta\in\Lambda} r_n(l,\theta)\\ &= \inf_l \sup_\theta \frac{1}{n} E_\theta\left[\log \frac{p_\theta(x^n)}{2^{-l(x^n)}}\right]\\ &= \inf_Q \sup_\theta \frac{1}{n} D(P_\theta \| Q),\end{aligned}$$

where Q is the prior induced by the coding length function l.

Minimal redundancy

Note that
Equation:

$$\begin{aligned}\forall w, l, \quad \sup_\theta r_n(l,\theta) &\geq \int_\Lambda w(d\theta) r_n(l,\theta)\\ &\geq \inf_{l\in c_n} \int_\Lambda w(d\theta) r_n(l,\theta).\end{aligned}$$

Therefore,
Equation:

$$R_n^+ = \inf_l \sup_\theta r_n(l,\theta) \geq \sup_w \inf_l \int_\Lambda w(d\theta) r_n(l,\theta) = R_n^-.$$

In fact, Gallager showed that $R_n^+ = R_n^-$. That is, the min-max and max-min redundancies are equal.

Let us revisit the Bernoulli source p_θ where $\theta \in \Lambda = [0,1]$. From the definition of [link], which relies on a uniform prior for the sources, i.e., $w(\theta) = 1, \forall \theta \in \Lambda$, it can be shown that there there exists a universal code with length function l such that
Equation:

$$E_\theta\left[l\left(x^n\right)\right] \leq nE_\theta\left[h_2\left(\frac{n_x\left(1\right)}{n}\right)\right] + \log\left(n+1\right) + 2,$$

where $h_2\left(p\right) = -p\log\left(p\right) - \left(1-p\right)\log\left(1-p\right)$ is the binary entropy. That is, the redundancy is approximately $\log\left(n\right)$ bits. Clarke and Barron [link] studied the weighting approach,
Equation:

$$p\left(x\right) = \int_\Lambda dw\left(\theta\right) p_\theta\left(x\right),$$

and constructed a prior that achieves $R_n^- = R_n^+$ precisely for memoryless sources.

Theorem 5 **[link]** For memoryless source with an alphabet of size r, $\theta = (p(0), p(1), \cdots, p(r-1))$,
Equation:

$$nR_n^-\left(w\right) = \frac{r-1}{2}\log\left(\frac{n}{2\pi e}\right) + \int_\Lambda w\left(d\theta\right)\log\left(\frac{\sqrt{|I(\theta)|}}{w(\theta)}\right) + O_n\left(1\right),$$

where $O_n\left(1\right)$ vanishes uniformly as $n \to \infty$ for any compact subset of Λ, and
Equation:

$$I\left(\theta\right) \triangleq E\left[\left(\frac{\partial \ln p_\theta\left(x_i\right)}{\partial\theta}\right)\left(\frac{\partial \ln p_\theta\left(x_i\right)}{\partial\theta}\right)^T\right]$$

is Fisher's information.

Note that when the parameter is sensitive to change we have large $I(\theta)$, which increases the redundancy. That is, good sensitivity means bad universal compression.

Denote

Equation:

$$J\left(\theta\right)=\frac{\sqrt{|I(\theta)|}}{\int_{\Lambda}\sqrt{|I(\theta')|}d\theta'},$$

this is known as *Jeffrey's prior*. Using $w(\theta)=J(\theta)$, it can be shown that $R_n^-=R_n^+$.

Example:
Let us derive the Fisher information $I(\theta)$ for the Bernoulli source,

Equation:

$$\begin{aligned}
& p_\theta\left(x\right)=\theta^{n_x(1)}\cdot\left(1-\theta\right)^{n_x(0)} \\
\Rightarrow\quad & \ln p_\theta\left(x\right)=n_x\left(1\right)\ln\theta+n_x\left(0\right)\ln\left(1-\theta\right) \\
\Rightarrow\quad & \frac{\partial\ln p_\theta\left(x\right)}{\partial\theta}=n_x\left(1\right)\frac{1}{\theta}-n_x\left(0\right)\frac{1}{1-\theta} \\
\Rightarrow\quad & \left(\frac{\partial\ln p_\theta\left(x\right)}{\partial\theta}\right)^2=\frac{n_x^2\left(1\right)}{\theta^2}+\frac{n_x^2\left(0\right)}{\left(1-\theta\right)^2}-\frac{2n_x\left(1\right)n_x\left(0\right)}{\theta(1-\theta)} \\
\Rightarrow\quad & E\left[\left(\frac{\partial\ln p_\theta\left(x\right)}{\partial\theta}\right)^2\right]=\frac{\theta}{\theta^2}+\frac{1-\theta}{\left(1-\theta\right)^2}-\frac{2}{\theta(1-\theta)}E\left[n_x\left(1\right)n_x\left(0\right)\right] \\
& =\frac{1}{\theta}+\frac{1}{1-\theta}-0 \\
& =\frac{1}{\theta(1-\theta)}.
\end{aligned}$$

Therefore, the Fisher information satisfies $I\left(\theta\right)=\frac{1}{\theta(1-\theta)}$.

Example:
Recall the Krichevsky–Trofimov coding, which was mentioned in [link].
Using the definition of Jeffreys' prior [link], we see that $J\left(\theta\right)\propto\frac{1}{\sqrt{\theta(1-\theta)}}$.
Taking the integral over Jeffery's prior,

Equation:

$$\begin{aligned} p_J\left(x^n\right) &= \int_0^1 c\frac{d\theta}{\sqrt{\theta(1-\theta)}}\theta^{n_x(1)}(1-\theta)^{n_x(0)} \\ &= c\int_0^1 \theta^{n_x(1)-\frac{1}{2}}(1-\theta)^{n_x(0)-\frac{1}{2}}d\theta \\ &= \frac{\Gamma\left(n_x\left(0\right)+\frac{1}{2}\right)\Gamma\left(n_x\left(1\right)+\frac{1}{2}\right)}{\pi\Gamma(n+1)}, \end{aligned}$$

where we used the gamma function. It can be shown that
Equation:

$$p_J\left(x^n\right)=\prod_{t=0}^{n}p_J\left(x_{t+1}\middle|x_1^t\right),$$

where
Equation:

$$\begin{aligned} p_J\left(x_{t+1}\middle|x_1^t\right) &= \frac{p_J\left(x_1^{t+1}\right)}{p_J\left(x_1^t\right)}, \\ p_J\left(x_{t+1}=0\middle|x_1^t\right) &= \frac{n_x^t\left(0\right)+\frac{1}{2}}{t+1}, \\ p_J\left(x_{t+1}=1\middle|x_1^t\right) &= \frac{n_x^t\left(1\right)+\frac{1}{2}}{t+1}. \end{aligned}$$

Similar to before, this universal code can be implemented sequentially. It is due to Krichevsky and Trofimov [link], its redundancy satisfies Theorem 5 by Clarke and Barron [link], and it is commonly used in universal lossless compression.

Rissanen's bound

Let us consider – on an intuitive level – why $C_n \approx \frac{r-1}{2}\frac{\log(n)}{n}$. Expending $\frac{r-1}{2}\log(n)$ bits allows to differentiate between $\left(\sqrt{n}\right)^{r-1}$ parameter vectors. That is, we would differentiate between each of the $r-1$ parameters with $\sqrt{n}$ levels. Now consider a Bernoulli RV with (unknown) parameter θ.

One perspective is that with n drawings of the RV, the standard deviation in the number of 1's is $O\left(\sqrt{n}\right)$. That is, $\sqrt{n}$ levels differentiate between parameter levels up to a resolution that reflects the randomness of the experiment.

A second perspective is that of coding a sequence of Bernoulli outcomes with an imprecise parameter, where it is convenient to think of a universal code in terms of first quantizing the parameter and then using that (imprecise) parameter to encode the input x. For the Bernoulli example, the *maximum likelihood* parameter θ_{ML} satisfies
Equation:

$$\theta_{ML} = \text{argmax}\left\{\theta^{n_x(1)}(1-\theta)^{n_x(0)}\right\},$$

and plugging this parameter $\theta = \theta_{ML}$ into $p_\theta(x)$ minimizes the coding length among all possible parameters, $\theta \in \Lambda$. It is readily seen that
Equation:

$$\theta_{ML} = \frac{n_x(1)}{n}.$$

Suppose, however, that we were to encode with $\theta' = \theta_{ML} + \Delta$. Then the coding length would be

Equation:

$$l_\theta\left(x\right)=-\log\left(\left(\theta'\right)^{n_x(1)}\left(1-\theta'\right)^{n_x(0)}\right).$$

It can be shown that this coding length is suboptimal w.r.t. $l_{\theta_{ML}}\left(x\right)$ by $n\cdot O\left(\Delta^2\right)$ bits. Keep in mind that doubling the number of parameter levels used by our universal encoder requires an extra bit to encode the extra factor of 2 in resolution. It makes sense to expend this extra bit only if it buys us at least one other bit, meaning that $n\cdot O\left(\Delta^2\right)=1$, which implies that we encode θ_{ML} to a resolution of $1/\sqrt{n}$, corresponding to $O\left(\sqrt{n}\right)$ levels. Again, this is a redundancy of $\approx\frac{1}{2}\log\left(n\right)$ bits per parameter.

Having described Rissanen's result intuitively, let us formalize matters. Consider $\{p_\theta,\theta\in\Lambda\}$, where $\Lambda\subset\mathbb{R}^K$ is a compact set. Suppose that there exists an estimator $\hat{\theta}$ such that
Equation:

$$\forall n\geq n\left(c\right):\quad p_\theta\left\{\parallel\hat{\theta}\left(x^n\right)-\theta\parallel>\frac{c}{\sqrt{n}}\right\}\leq\delta\left(c\right),$$

where $\lim_{c\to\infty}\delta\left(c\right)=0$. Then we have the following converse result.

Theorem 6 (Converse to universal coding [link]) Given a parametric class that satisfies the above condition [link], for all $\epsilon>0$ and all codes l that do not know θ,
Equation:

$$r_n\left(l,\theta\right)\geq\left(1-\epsilon\right)\frac{K}{2}\frac{\log\left(n\right)}{n},$$

except for a class of θ in $B_\epsilon\left(n\right)\subseteq\Lambda$ whose Lebesgue volume shrinks to zero as n increases.

That is, a universal code cannot compress at a redundancy substantialy below $\frac{1}{2}\log\left(n\right)$ bits per parameter. Rissanen also proved the following achievable result in his seminal paper.

Theorem 7 (Achievable to universal coding [link]) If $p_\theta(x)$ is twice differentiable in θ for every x^n, then there exists a universal code such that $\forall\theta\in\Lambda : r_n(l,\theta)\le(1+\epsilon)\frac{K}{2}\frac{\log(n)}{n}$.

Universal coding for piecewise i.i.d. sources

We have emphasized stationary parametric classes, but a parametric class can be nonstationary. Let us show how universal coding can be achieved for some nonstationary classes of sources by providing an example. Consider $\Lambda=\{0,1,\ldots,n\}$ where
Equation:

$$p_\theta(x^n)=Q_1\left(x_1^\theta\right)\cdot Q_2\left(x_{\theta+1}^n\right),$$

where Q_1 and Q_2 are both know i.i.d. sources. This is a *piecewise i.i.d. source*; in each segment it is i.i.d., and there is an abrupt transition in statistics when the first segment ends and the second begins.

Here are two approaches to coding this source.

1. Encode the best index θ_{ML} using $\lceil\log(n+1)\rceil$ bits, then encode $p_{\theta_{ML}}(x^n)$. This is known as *two-part code* or *plug-in*; after encoding the index, we plug the best parameter into the distribution. Clearly,
   Equation:

$$\begin{aligned} l(x) &= \min_{0\le\theta\le n}\lceil-\log p_\theta(x)\rceil+\lceil\log(n+1)\rceil \\ &\le -\log p_\theta(x)+\log(n+1)+2. \end{aligned}$$

2. The second approach is a *mixture*, we allocate weights for all possible parameters,
   Equation:

$$\begin{aligned} l(x) &= -\log\left(\frac{1}{n+1}\sum_{i=0}^{n} p_i\left(x^n\right)\right) \\ &< -\log\left(\frac{1}{n+1} p_{\theta_{ML}}\left(x^n\right)\right) \\ &= -\log\left(p_{\theta_{ML}}\left(x\right)\right) + \log\left(n+1\right). \end{aligned}$$

Merhav [link] provided redundancy theorems for this class of sources. Algorithmic approaches to the mixture appear in Shamir and Merhav [link] and Willems [link].

The theme that is common to both approaches, the plug-in and the mixture, is that they lose approximately $\log\left(n\right)$ bits in encoding the location of the transition. Indeed, Merhav showed that the penalty for each transition in universal coding is approximately $\log\left(n\right)$ bits [link]. Intuitively, the reason that the redundancy required to encode the location of the transition is larger than the $\frac{1}{2}\log\left(n\right)$ from Rissanen [link] is because the location of the transition must be described precisely to prevent paying a big coding length penalty in encoding segments using the wrong i.i.d. statistics. In contrast, in encoding our Bernoulli example an imprecision of $\frac{1}{\sqrt{n}}$ in encoding θ_{ML} in the first part of the code yields only an $O(1)$ bit penalty in the second part of the code.

It is well known that mixtures out-compress the plug-in. However, in many cases they do so by only a small amount per parameter. For example, Baron et al. showed that the plug-in for i.i.d. sources loses approximately 1 bit per parameter w.r.t. the mixture.

Semi-predictive approach

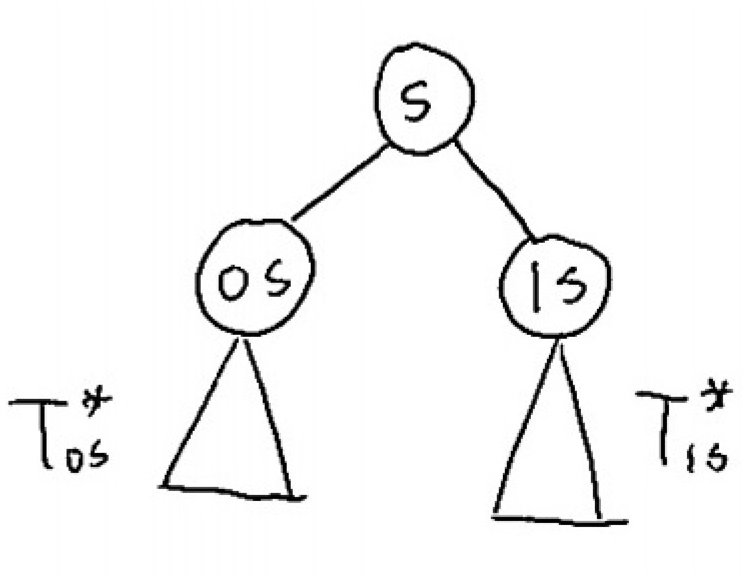
Recall that a context tree source is similar to a Markov source, where the number of states is greatly reduced. Let T be the set of leaves of a context tree source, then the redundancy is
Equation:

$$r \lesssim \frac{|T|(r-1)}{2}\left(\log\left(\frac{n}{|T|}\right)+O\left(1\right)\right),$$

where $|T|$ is the number of leaves, and we have $\log\left(\frac{n}{|T|}\right)$ instead of $\log\left(n\right)$, because each state generated $\frac{n}{|T|}$ symbols, on average. In contrast, the redundancy for a Markov representation of the tree source T is much larger. Therefore, tree sources are greatly preferable in practice, they offer a significant reduction in redundancy.

How can we compress universally over the parametric class of tree sources? Suppose that we knew T, that is we knew the set of leaves. Then we could process x sequentially, where for each x_i we can determine what state its context is in, that is the unique suffix of x_1^{i-1} that belongs to the set of leaf labels in T. Having determined that we are in some state s, $\Pr\left(x_i=0\middle|s,x_1^{i-1}\right)$ can be computed by examining all previous times that we were in state s and computing the probability with the Krichevsky-Trofimov approach based on the number of times that the following symbol (after s) was 0 or 1. In fact, we can store symbol counts $n_x\left(s,0\right)$ and $n_x\left(s,1\right)$ for all $s\in T$, update them sequentially as we process x, and compute $\Pr\left(x_i=0\middle|s,x_1^{i-1}\right)$ efficiently. (The actual translation to bits is performed with an arithmetic encoder.)

While promising, this approach above requires to know T. How do we compute the optimal T^* from the data?



Tree pruning in the semi-predictive approach.

Semi-predictive coding: The *semi-predictive* approach to encoding for context tree sources [link] is to scan the data twice, where in the first scan we estimate T^* and in the second scan we encode x from T^*, as described above. Let us describe a procedure for computing the optimal T^* among tree sources whose depth is bounded by D. This procedure is visualized in [link]. As suggested above, we count $n_x(s,a)$, the number of times that each possible symbol appeared in context s, for all $s \in \alpha^D, a \in \alpha$. Having computed all the symbol counts, we process the depth-D tree in a bottom-top fashion, from the leaves to the root, where for each internal node s of the tree (that is, $s \in \alpha^d$ where $d < D$), we track T_s^*, the optimal tree structure rooted at s to encode symbols whose context ends with s, and $\text{MDL}(s)$ the minimum description lengths (MDL) required for encoding these symbols.

Without loss of generality, consider the simple case of a binary alphabet $\alpha = \{0,1\}$. When processing s we have already computed the symbol counts $n_x(0s,0)$ and $n_x(0s,1)$, $n_x(1s,0)$, $n_x(1s,1)$, the optimal trees

T_{0s}^{*} and T_{1s}^{*}, and the minimum description lengths (MDL) $\mathrm{MDL}(0s)$ and $\mathrm{MDL}(1S)$. We have two options for state s.

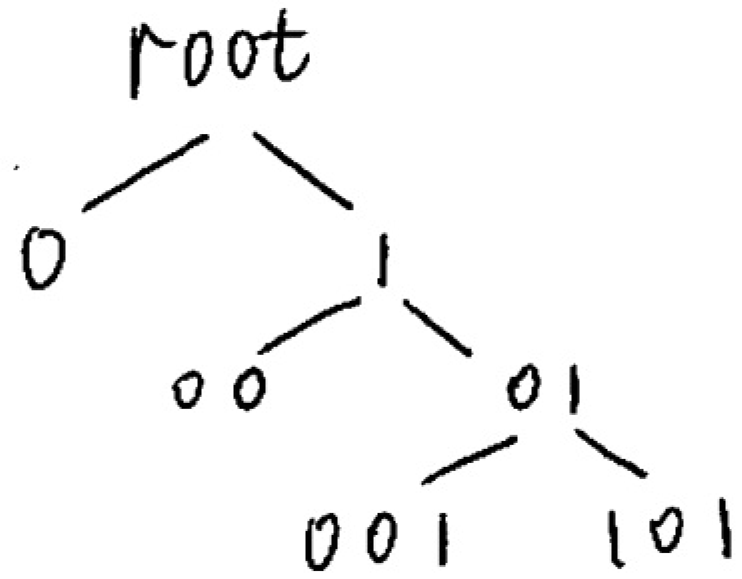
1. *Keep*T_{0S}^{*} and T_{1S}^{*}. The coding length required to do so is $\mathrm{MDL}(0S) + \mathrm{MDL}(1S) + 1$, where the extra bit is spent to describe the structure of the maximizing tree.
2. *Merge* both states (this is also called *tree pruning*). The symbol counts will be $n_x(s, \alpha) = n_x(0s, \alpha) + n_x(1s, \alpha), \alpha \in \{0, 1\}$, and the coding length will be
   Equation:

   $$\mathrm{KT}(n_x(s, 0), n_x(s, 1)) + 1,$$

   where $\mathrm{KT}(\cdot, \cdot)$ is the Krichevsky-Trofimov length [link], and we again included an extra bit for the structure of the tree.

We note in past that there is no need to spend a bit to encode leaves of depth D. To see this, consider a procedure for encoding the structure of a tree:

Example:
Consider the tree sourced depicted in [link]. In order to encode the structure of this tree, we will utilize the following procedure. (Such a procedure has appeared, for example, in [link].)



Tree used in [link] to demonstrate how the structure of the source is encoded.

```
Start from root. [procedure(root)]
1. If node $S$ is of depth $D$ (maximum), then return.
2. If node $S$ is internal node, then {
    encode 0
    procedure(0S)
    procedure(1S)
} else encode 1.
3. return.
```

Let us now simulate the procedure, the procedure will traverse through the following states of the tree in [link] while outputting the corresponding bits.

Source	root	0	1	01	001	101	11
Encoded symbol	0	1	0	0			1

Returning to tree pruning, following [link] we see that we must initialize $\mathrm{MDL}\,(s) = \mathrm{KT}(n_x\,(s,0), n_x\,(s,1))$ for s of full depth $|s| = D$ without the extra bit.

At the end of the pruning procedure, $T^*_{\{\}}$ the maximizing tree for the root, will be the optimal tree for universal coding.

Burrows wheeler Transform

The Burrows Wheeler transform (BWT) was proposed by Burrows and Wheeler in 1994 [link] (see also the analysis by Effros et al. [link] and references therein). It is an invertible permutation sort that sorts symbols according to their contexts. That way, the symbols that were generated by the same state of the context tree are grouped together, which as we will see is advantageous.

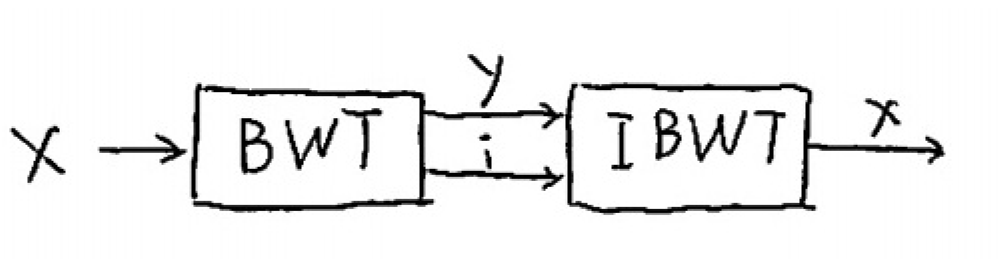
To compute the BWT, we first compute all cyclical shifts of the input x. Next, we sort the cyclical shifts. The output of the BWT consists of y, the last column of the matrix of sorted shifts, and i the index of the original version. We illustrate with an example.

Example:
Consider the input $x = banana$. First, we compute the cyclic shifts and their sorts.

All Shifts	Sorted
banana	abanan
abanan	anaban
nabana	ananab
anaban	banana
nanaba	nabana
ananab	nanaba

The output of the BWT consists of $y = nnbaaa$, the last column of the matrix of sorted shifts (to the right), and the index $i = 4$ containing the original input.

Interestingly, we can recover x from y and i. Seeing that y is structured and thus quite compressible, the BWT can be used as a compression system; a building block that illustrates such a system appears in [link].



Typical compression system using the Burrows Wheeler transform [link].

To see that the BWT is invertible, let us work out how to do this by continuing our example.

Example:
In the matrix of sorted shifts, column 1 is a sorted version of column n, which we know.

Column 1	Column n
a	n
a	n
a	b
b	a
n	a
n	a

Now take column n and put it before column 1:

Column n	Column 1

n	a
n	a
b	a
a	b
a	n
a	n

We now sort these rows, which each consist of 2 symbols: ab, an, an, ba, na, and na. Now fill column 2 of the sorted shifts matrix accordingly.

Columns 1–2	Column n
ab	n
an	n
an	b
ba	a
na	a
na	a

The entire matrix can be unraveled, and the row containing the original x is indexed by i.

What is the BWT good for? The key property of the BWT is that symbols generated by the same state are grouped together in y. To see this, note how the last column n can be rotated to a position to the left of column 1, and symbols that came before the same prefix appear together. (To bunch together symbols generated by the same suffix, we can reverse the order of symbols in x before running the BWT.) Therefore, y has the form of a piecewise i.i.d. sequence [link], where segments generated by the same state of the context tree are bunched together.

As a consequence of the structure of y, it is easy to see that it can be compressed with the following redundancy,
Equation:

$$r \lesssim \frac{|T|(r-1)}{2}\left(\log\left(\frac{n}{|T|}+O\left(1\right)\right)\right)+|T|\,\log\,(n),$$

where the new $|T|\,\log\,(n)$ term arises from coding the locations of transitions between segments (states of the tree) in the BWT output. Not only is the BWT convenient for compression, but it is amenable to fast computation. Both the BWT and its inverse can be implemented in $O(n)$ time. This combination of great compression and speed has made the BWT quite popular in compressors that have appeared since the late 1990s. For example, the bzip2 archiving package is very popular among network administrators.

That said, from a theoretical perspective the BWT suffers from an extraneous redundancy of $|T|\,\log\,(n)$ bits. Until this gap was resolved, the theoretical community still preferred the semi-predictive method or another approach based on mixtures.

Semi-predictive coding using the BWT

Another approach for using the BWT is to use y only for learning the MDL tree source T^*. To do so, note that when the BWT is run, it is possible to track the correspondences between contexts and segments of the BWT output. Therefore, information about per-segment symbol count is available, and can be easily applied to perform the tree pruning procedure that we have seen. Not only that, but some BWT computation algorithms (e.g., suffix tree approaches) maintain this information for all context depths and not just bounded D. In short, the BWT allows to compute the minimizing tree T^* in linear time [link].

Given the minimizing tree T^*, it is not obvious how to determine which state generated each character of y (respectively, x) in linear time. It has been shown by Martín et al. [link] that this step can also be performed in linear time by developing a state machine whose states include the leaves of T^*. The result is a two part code, where the first part computes the optimal T^* via BWT, and the second part actually compresses x by tracking which state of T^* generated each of the symbols. To summarize, we have a linear complexity algorithm for compressing and decompressing a source while achieving the redundancy bounds for the class of tree sources.

Context Tree Weighting

We discussed in [link] for the problem of encoding a transition between two known i.i.d. distributions that
Equation:

$$\frac{1}{n}\sum_{i=1}^{n} p_{\theta_i}(x) > \frac{1}{n}\max_i \{p_{\theta_i}(x)\}.$$

Therefore, a mixture over all parameter values yields a greater probability (and thus lower coding length) than the maximizing approach. Keep in mind that finding the optimal MDL tree source T^* is analogous to the plug-in approach, and it would reduce the coding length if we could assign the probability as a mixture over all possible trees, where we assign trees with fewer leaves a greater weight. That is, ideally we want to implement
Equation:

$$\Pr(x) = \sum_T 2^{-|\text{code}(T)|} \cdot p_T(x),$$

where $|\mathrm{code}(T)|$ is the length of the encoding procedure that we discussed for the tree structure T, and $p_T(x)$ is the probability for the sequence x under the model T.

Willems et al. showed how to implement such a mixture in a simple way over the class of tree sources of bounded depth D. As before, the algorithm proceeds in a bottom up manner from leaves toward the root. At leaves, the probability p_s assigned to symbols that were generated within that context s is the Krichevsky-Trofimov probability, $p_{KT}(s,x)$ [link]. For s that is an internal node whose depth is less than D, the approach by Willems et al. [link] is to mix (*i*) the probabilities of keeping the branches for 0s and 1s and (*ii*) pruning,
Equation:

$$p_s = \frac{1}{2} p_K T(s,x) + \frac{1}{2} p_{0s} \cdot p_{1s}.$$

It can be shown that this simple formula allows to implement a mixture over the class of bounded depth context tree sources, thus reducing the coding length w.r.t. the semi-predictive approach.

In fact, Willems later showed how to extend the context tree weighting (CTW) approach to tree sources of unbounded depth [link]. Unfortunately, while the basic bounded depth CTW has complexity that is comparable to the BWT, the unbounded CTW has potentially higher complexity.

Universal lossy compression

Consider $x \in \alpha^n$. The goal of lossy compression [link] is to describe $\widehat{x}$, also of length n but possibly defined over another reconstruction alphabet $\widehat{\alpha} \neq \alpha$, such that the description requires few bits and the distortion
Equation:

$$\bar{d}\left(x, \widehat{x}\right) = \frac{1}{n} \sum_{i=1}^{n} d\left(x_i, \widehat{x}_i\right)$$

is small, where $d(\cdot, \cdot)$ is some distortion metric. It is well known that for every $d(\cdot, \cdot)$ and distortion level D there is a minimum rate $R(D)$, such that $\widehat{x}$ can be described at rate $R(D)$. The rate $R(D)$ is known as the rate distortion (RD) function, it is the fundamental information theoretic limit of lossy compression [link], [link].

The invention of lossy compression algorithms has been a challenging problem for decades. Despite numerous applications such as image compression [link], [link], video compression [link], and speech coding [link], [link], [link], there is a significant gap between theory and practice, and these practical lossy compressors do not achieve the RD function. On the other hand, theoretical constructions that achieve the RD function are impractical.

A promising recent algorithm by Jalali and Weissman [link] is universal in the limit of infinite runtime. Its RD performance is reasonable even with modest runtime. The main idea is that the distortion version $\widehat{x}$ of the input x can be computed as follows,
Equation:

$$\widehat{x} = \underset{w^n \in \alpha^n}{\operatorname{argmin}} \left\{ H_k\left(w^n\right) - \beta \bar{d}\left(x^n, w^n\right) \right\},$$

where $\beta < 0$ is the slope at the particular point of interest in the RD function, and $H_k\left(w^n\right)$ is the *empirical conditional entropy* of order k,
Equation:

$$H_k\left(w^n\right) \triangleq -\frac{1}{n} \sum_{a, u^k} n_w\left(u^k, a\right) \log\left(\frac{n_w\left(u^k, a\right)}{\sum_{a' \in \alpha} n_w\left(u^k, a'\right)}\right),$$

where u^k is a context of order k, and as before $n_w\left(u^k, a\right)$ is the number of times that the symbol a appears following a context u^k in w^n. Jalali and Weissman proved [link] that when $k = o(\log(n))$, the RD pair $\left(H_k\left(\widehat{x}^n\right), \bar{d}\left(x^n, \widehat{x}^n\right)\right)$ converges to the RD function asymptotically in n. Therefore, an excellent lossy compression technique is to compute $\widehat{x}$ and then compress it. Moreover, this compression can be universal. In particular, the choice of context order $k = o(\log(n))$ ensures that universal compressors for context tress sources can emulate the coding length of the empirical conditional entropy $\widehat{H}_k\left(\widehat{x}^n\right)$.

Despite this excellent potential performance, there is still a tremendous challenge. Brute force computation of the globally minimum energy solution $\widehat{x^n}$ involves an exhaustive search over exponentially many sequences and is thus infeasible. Therefore, Jalali and Weissman rely on *Markov chain Monte Carlo* (MCMC) [link], which is a stochastic relaxation approach to optimization. The crux of the matter is to define an energy function,
Equation:

$$\epsilon\left(w^n\right) = H_k\left(w^n\right) - \beta d\left(x^n, w^n\right).$$

The Boltzmann probability mass function (pmf) is
Equation:

$$f_s\left(w^n\right) \triangleq \frac{1}{Z_t} \exp\left\{-\frac{1}{t}\varepsilon\left(w^n\right)\right\},$$

where $t > 0$ is related to temperature in simulated annealing, and Z_t is the normalization constant, which does not need to be computed.

Because it is difficult to sample from the Boltzmann pmf [link] directly, we instead use a *Gibbs sampler*, which computes the marginal distributions at all n locations conditioned on the rest of w^n being kept fixed. For each location, the Gibbs sampler resamples from the distribution of w_i conditioned on $w^{n\backslash i} \triangleq \{w_n : \ n \neq i\}$ as induced by the joint pmf in [link], which is computed as follows,
Equation:

$$\begin{aligned} \Pr\left(w_i = b \middle| w^{n\backslash i}\right) &= \frac{\Pr\left(w_i = b, w^{n\backslash i}\right)}{\Pr\left(w^{n\backslash i}\right)} \\ &= \frac{\Pr\left(w_i = b, w^{n\backslash i}\right)}{\sum_{b' \in \widehat{\alpha}} \Pr\left(w_i = b', w^{n\backslash i}\right)} \\ &= \frac{\frac{1}{Z_t} \exp\left\{-\frac{1}{t}\epsilon\left(w_i = b, w^{n\backslash i}\right)\right\}}{\sum_{b'} \frac{1}{Z_t} \exp\left\{-\frac{1}{t}\epsilon\left(w_i = b', w^{n\backslash i}\right)\right\}}. \end{aligned}$$

We can now write,
Equation:

$$\epsilon\left(y_i = b, y^{n\backslash i}\right) = \epsilon\left(y_i = a\right) + \mathit{\Delta}H\left(b, a\right) - \beta\mathit{\Delta}d\left(b, a\right),$$

where $\varDelta H_k\left(y^{i-1}by_{i+1}^n, a\right)$ is the change in $H_k\left(w^n\right)$ when $y_i = a$ is replaced by b, and $\varDelta d\left(b, a, x_i\right) = d\left(b, x_i\right) - d\left(a, x_i\right) = \left(b - x_i\right)^2 - \left(a - x_i\right)^2$ is the change in distortion. Combining [link] and [link],
Equation:

$$\begin{aligned}\Pr\left(w_i = b \middle| w^{n\backslash i}\right) &= \frac{\exp\left\{-\frac{1}{t}\left[\epsilon\left(a\right) + \varDelta H\left(b, a\right) - \beta \varDelta d\left(b, a\right)\right]\right\}}{\sum_{b'} \exp\left\{-\frac{1}{t}\left[\epsilon\left(a\right) + \varDelta H\left(b', a\right) - \beta \varDelta d\left(b', a\right)\right]\right\}} \\ &= \frac{\exp\left\{-\frac{1}{t} \cdot 0\right\}}{\sum_{b'} \exp\left\{-\frac{1}{t}\left[\varDelta H\left(b', a\right) - \beta \varDelta d\left(b', a\right) - \varDelta H\left(b, a\right) + \beta \varDelta d\left(b, a\right)\right]\right\}} \\ &= \frac{1}{\sum_{b'} \exp\left\{-\frac{1}{t}\left[\varDelta H\left(b', b\right) - \beta \varDelta d\left(b', b\right)\right]\right\}}\end{aligned}$$

The maximum change in the energy within an iteration of MCMC algorithm is then bounded by
Equation:

$$\varDelta_k = \max_{1 \le i \le n} \max_{w^n \in \widehat{\alpha}^n} \max_{b, b' \in \widehat{\alpha}} \left| n \varDelta H_k\left(b, b'\right) + c_4 \varDelta d\left(b, b'\right)\right|.$$

We refer to the resampling from a single location as an iteration, and group the n possible locations into super-iterations.[footnote]
Baron and Weissman [link] recommend an ordering where each super-iteration scans a permutation of all n locations of the input, because in this manner each location is scanned fairly often. Other orderings are possible, including a completely random order.

During the simulated annealing, the temperature t is gradually increased, where in super-iteration i we use $t = O(1/\log(i))$ [link], [link]. In each iteration, the Gibbs sampler modifies w^n in a random manner that resembles heat bath concepts in statistical physics. Although MCMC could sink into a local minimum, we decrease the temperature slowly enough that the randomness of Gibbs sampling eventually drives MCMC out of the local minimum toward the set of minimal energy solutions, which includes $\widehat{x^n}$, because low temperature t favors low-energy w^n. Pseudo-code for our encoder appears in Algorithm 1 below.

Algorithm 1: Lossy encoder with fixed reproduction alphabetInput:$x^n \in \alpha^n$, $\widehat{\alpha}$, β, c, rOutput: bit-stream Procedure:

```
1. Initialize $w$ by quantizing $x$ with $\widehat{\alpha}$
2. Initialize $n_w(\cdot,\cdot)$ using $w$
3. **for** $r = 1$ to $R$ **do** // *super-iteration*
4.     $t_r \leftarrow \frac{cn\varDelta_k}{\log(r)}$ // *temperature*
5.     Draw permutation of numbers $\{1, ..., n\}$ at random
6.     **for** $r' = 1$ to $n$ **do**
7.       Let $j$ be component $t'$ in permutation
8.       Generate new $w_j$ using $f_s\left(w_j = \cdot \middle| w^{n\backslash j}\right)$
```

```
9.        Update $n_w(\cdot,\cdot)[\cdot]$
10. Apply CTW to $w^n$ // *compress outcome*
```

Computational issues

Looking at the pseudo-code, it is clear that the following could be computational bottlenecks:

1. Initializing $n_w(\cdot,\cdot)$ - a naive implementation needs to scan the sequence w^n (complexity $O(n)$) and initialize a data structure with $|\widehat{\alpha}|^{K+1}$ elements. Unless $K \lesssim \log_{|\widehat{\alpha}|}(n)$, this is super linear in n. Therefore, we recall that $K = o(\log(n))$, and initializing $n_w(\cdot,\cdot)$ requires linear complexity $O(n)$.
2. The inner loop is run rn times, and each time computing $\Pr\left(w_j = b \middle| w^{n\setminus j}\right)$ for all possible $b \in \widehat{\alpha}$ might be challenging. In particular, let us consider computation of Δd and ΔH.

    1. Computation of Δd requires constant time, and is not burdensome.
    2. Computation of ΔH requires to modify the symbol counts for each context that was modified. A key contribution by Jalali and Weissman was to recognize that the array of symbol counts, $n_w(\cdot,\cdot)$, would change in $O(k)$ locations, where k is the context order. Therefore, each computation of ΔH requires $O(k)$ time. Seeing that $|\widehat{\alpha}|$ such computations per iteration are needed, and there are rn iterations, this is $O\left(krn|\widehat{\alpha}|\right)$.
    3. Updating $n_w(\cdot,\cdot)$ after w_j is re-sampled from the Boltzmann distribution also requires $O(k)$ time. However, this step is performed only once per iteration, and not $|\widehat{\alpha}|$ times. Therefore, this step requires less computation than step (b).

Lossy compression of an analog input

So far, we have described the approach by Jalali and Weissman to compress $x^n \in \alpha^n$. Suppose instead that $x \in \mathbb{R}^n$ is analog. Seeing that MCMC optimizes w^n over a discrete alphabet, it would be convenient to keep doing so. However, because $\widehat{\alpha}$ is finite, and assuming that square distortion is used, i.i., $d(x_i, w_i) = (x_i - w_i)^2$, we see that the distortion could be large.

Fortunately, Baron and Weissman showed [link] that the following quantizer has favorable properties,
Equation:

$$\widehat{\alpha} = \left\{ \frac{-\gamma^2}{\gamma}, \frac{-\gamma^2+1}{\gamma}, \cdots, 0, \frac{1}{\gamma}, \cdots, \frac{+\gamma^2}{\gamma} \right\},$$

where γ is an integer that grows with n. That is, as γ grows the set $\widehat{\alpha}$ of reproduction levels quantizers a wider internal more finely. Therefore, we have that the probability that some x_i is an outlier, i.e., $|x_i| < \gamma$, vanishes with n, and the effect of outliers on $\overline{d}(x, \overline{x})$ vanishes. Moreover, because the internal is sampled more finely as γ increases with n, this quantizer can emulate any codebook with continuous levels, and so in the limit its RD performance converges to the RD function.

Rate distortion performance

To evaluate the RD performance, we must first define the RD function. Consider a source X that generates a sequence $x^n \in \mathbb{R}^n$. A *lossy encoder* is a mapping $e : \mathbb{R}^n \rightarrow \mathscr{C}$, where $\mathscr{C} \subset \mathbb{R}^n$ is a codebook that contains a set of codewords in $\mathbb{R}^n$. Let $\mathscr{C}^n(x, D)$ be the smallest cardinality codebook for inputs of length n generated by X such that the expected per-symbol distortion between the input x^n and the codeword $e(x) \in \mathscr{C}^n(x, D)$ is at most D. The rate $R^n(x, D)$ is the per-symbol log-cardinality of the codebook,
Equation:

$$R^n(x, D) = \frac{1}{n}\log\left(|\mathscr{C}^n(x, D)|\right).$$

This is an operational definition of the RD function in terms of the best block code. In contrast, it can be shown that the RD function is equal to a minimization over mutual information, yielding a different flavor of definition [link], [link].

Having defined the RD function, we can describe the RD performance of the compression algorithm by Baron and Weissman for continuous valued inputs.

Theorem 8 Consider square error distortion, $d\left(x_i, \widehat{x}_i\right) = \left(x_i - \widehat{x}_i\right)^2$, let X be a finite variance stationary and ergodic source with RD function $R(X, D)$, and use the MCMC algorithm with data independent reproduction alphabet $\widehat{\alpha}$ and temperature that decays sufficiently slowly. Let w_r^n be the approximation to x^n after r super-iterations. Then the length of context tree weighting (CTW) applied to w_r^n converges as follows,
Equation:

$$\lim_{n\rightarrow\infty}\lim_{r\rightarrow\infty} E\left[\frac{1}{n}|CTW\left(W_r^n\right)| - \beta\overline{d}\left(x^n, W_r^n\right)\right]\xrightarrow{n\rightarrow\infty}\min_{D\geq 0}\left[R(X, D) - \beta D\right].$$

Remark 1 Let us make some comments.

- This result implies that CTW is used to compress w_r^n after it has been generated by MCMC. Instead of CTW, other universal compression approaches can be used.
- The same result can be proved for l_p distortion metrics, and not just l_2.
- A result with similar flavor was proved by Jalali and Weissman for discrete alphabets [link].

Adaptive reproduction levels: The above algorithm is promising from a theoretical perspective, but is of limited practical interest. In order to approach the RD function closely, $\widehat{\alpha}$ may need to be large, which slows down the algorithm.

Our approach to overcome the large alphabet is inspired by the work by Rose [link], who showed that in many cases the optimal RD codebook uses a small discrete-valued reproduction alphabet. This runs counter to our intuition that a continuous reproduction alphabet is needed to compress a continuous valued source. Therefore, we propose an algorithm that allows for reduction of the alphabet size while supporting adaptive reproduction levels.

We map the input x^n to a sequence z^n over a finite alphabet, where actual numerical values are obtained by a scalar function, $g(z_i)$. Ideally, $g(\cdot)$ should minimize distortion. Because we focus on square error distortion, the optimal $g^*(\cdot)$ is conditional expectation,
Equation:

$$g^*(\alpha) = E\left[x_i | z_i = \alpha\right] = \frac{\sum_{i:z_i=a} x_i}{\sum_{i:z_i=a_i} 1}.$$

Keep in mind that in an actual compression system, the encoder knows x^n, but the decoder does not. Therefore, the encoder must describe (a quantized version of) $g^*(\cdot)$ to the decoder. Putting aside these details, the main point is that the adaptive reproduction alphabet algorithm also acheives the RD function for stationary and ergodic sources while allowing a reduction in the size of the alphabet, thus accelerating the algorithm.

Universal signal reconstruction

Scalar channel: Consider a scalar channel, $y = x + z$, where $x, y, z \in \mathbb{R}^n$, x is generated by a stationalry and ergodic source X, and $z_i \sim N(0, 1)$ is zero mean unit norm Gaussian noise. To avoid taking the analysis toward continuous valued x, we can simplify the setting and temporarily consider discrete-valued x, y, and z; for example, we can consider a quantized version of the continuous problem.

Donoho [link] proposed the following reconstruction,
Equation:

$$\widehat{x} = \underset{ws.t.\|Y-w\|_2^2 \le n}{\text{argmin}} K(w),$$

where w is a possible reconstruction sequence, and $K(w)$ is the Kolmogorov complexity of w. Donoho calls this reconstruction algorithm the *Kolmogorov sampler*.

What is the Kolmogorov complexity [link], [link], [link] of the sequence w? It is the length of the shortest computer program that outputs w and then halts. One may ask what computer language we can use for the program; after all, it is easy to imagine some programming language being well-tuned to a specific sequence w. However, concepts such as universal Turing machines [link] can be used to put forward an abstract programming language that will run on any hypothetical computer. The key point is that the computer is limited to a state machine that processes an infinite-length tape, and one state machine can emulate another by programming a compiler onto the tape.

Is it practical to talk about the shortest program that outputs w and then halts? Not really, because we would need to run the Turing machine [link] (the computer) on an exponential number of programs. Moreover, it is well known that for some programs the Turing machine never halts (there are endless loops), and technical conditions such as time-out mechanisms will need to be added in. Therefore, computing the Kolmogorov complexity $K(w)$ is impractical [link], [link],

[link]. However, for w that was generated by a stationary ergodic source, it is possible to approximate $K(w)$ by $H_k(w)$, the empirical entropy [link], [link].

What is the performance of Donoho's Kolmogorov sampler? Donoho proved that $\widehat{x}$ has the following distribution, $\widehat{X} \sim p(X|Y)$ [link]. That is, $\widehat{x}$ is sampled from the posterior. Therefore, given y, we have that $\widehat{x}$ and x are both generated by the same distribution.

Consider now that $\widehat{x}_{MMSE}$, the minimum mean square error estimator, is the center of mass of the posterior. Seeing that $\widehat{x} - \widehat{x}_{MMSE}$ and $x - \widehat{x}_{MMSE}$ are orthogonal,
Equation:

$$E[\| x - \widehat{x} \|_2^2] = E[\| x - \widehat{x}_{MMSE} \|_2^2] + E[\| \widehat{x}_{MMSE} - \widehat{x} \|_2^2],$$

and so $E\left[\| x - \widehat{x} \|_2^2\right]$, the mean square error, is double the MMSE [link].

Let us pause to reflect about this result. When the SNR is high, i.e., $\| x \|_2^2 \gg \| z \|_2^2$, the MMSE should be rather low, and double the MMSE seems pretty good. On the other hand, when the SNR is low, the MMSE could be almost as large as $E\left[\| x \|_2^2\right]$, and double the MMSE could be larger – as much as twice larger – than $E\left[\| x \|_2^2\right]$. That is, gussing $\widehat{x} = E[x]$ could give better signal estimation performance than using the Kolmogorov sampler. This pessimistic result encourages us to search for better signal reconstruction methods.

Arbitrary channels: So far we considered the Kolmogorov sampler for the white scalar channel, $y = x + z$. Suppose instead that x is processed or measured by a more complicated system,
Equation:

$$y = J(x) + z.$$

Note that J is known, e.g., in a compressed sensing application [link], [link]J would be a known matrix. An even more involved system would be $y = (J \otimes x) \oplus z$, where $J \otimes x$ is application of a mapping to x, $J \otimes x \in \mathbb{R}^L$, and $\oplus z$ denotes application of a random noise operator to $J \otimes x$. To keep the presentation simple, we use the additive noise setting [link].

How can the Kolmogorov sampler [link] be applied to the additive noise setting? Recall that for the scalar channel, the Kolmogorov sampler minimizes $K(w)$ subject to $\| y - w \|_2^2 \leq n$. For the arbitrary mapping J with additive noise [link], this implies $\| y - J(w) \|_2^2 \leq n$. Therefore, we get
Equation:

$$\widehat{x} = \underset{s.t. \| Y - J(w) \|_2^2 \leq n}{\text{argmin}} K(w).$$

Another similar approach relies on optimization via Lagrange multipliers,
Equation:

$$\widehat{x} = \text{argmin } K(w) - \log_2 \left(f_z \left(y - J(w)\right)\right),$$

where the lagrange multiplier is 1, because both $K(w)$ and $\log_2 \left(f_z \left(y - J(w)\right)\right)$ are quantified in bits.

What is the performance of the Kolmogorov sampler for an arbitrary J? We speculate [link], [link] that $\widehat{x}$ is generated by the posterior, and so $E\left[\| x - \hat{\|}_2^2\right]$ is double the MMSE, where expectation is taken over the source X and noise z. These results remain to be shown rigorously.

Convergence of MCMC algorithm

We will now prove a substantial result – that the MCMC algorithm [link], [link], [link] converges to the globally minimal energy solution for the specific case of compressed sensing [link], [link]. An extension of this proof to arbitrary channels J is in progress.

If the operator J in [link] is a matrix, and we denote it by $\Phi \in \mathbb{R}^{m \times n}$ where $m \ll n$, then the setup is known as compressed sensing (CS) [link], [link] and the estimation problem is commonly referred to as recovery or reconstruction. By posing a sparsity or compressibility requirement on the signal and using it as a prior during recovery, it is indeed possible to accurately estimate x from y in the CS setting.

With the quantization alphabet definition in [link], $\widehat{\alpha}$ will quantize x to a greater resolution as N increases. We will show that under suitable conditions on f_X, performing maximum *a posteriori* (MAP) estimation over the discrete alphabet $\widehat{\alpha}$ asymptotically converges to the MAP estimate over the continuous distribution f_X. This reduces the complexity of the estimation problem from continuous to discrete.

We assume for exposition that we know the input statistics f_X. Given the measurements y, the MAP estimator for x has the form
Equation:

$$x_{MAP} \triangleq \underset{w}{\text{argmax}} f_X(w) f_{Y|X}(y|w).$$

Because z is i.i.d. Gaussian with mean zero and known variance σ_Z^2,
Equation:

$$f_{Y|X}(y|w) = c_1 e^{-c_2 \|y - \Phi w\|^2},$$

where $c_1 = \left(2\pi\sigma_Z^2\right)^{-M/2}$ and $c_2 = \frac{1}{2\sigma_Z^2}$ are constants and $\|\cdot\|$ denotes the Euclidean norm. Plugging into [link] and taking log likelihoods, we obtain
Equation:

$$x_{MAP} = \underset{w}{\text{argmin}} \Psi^X(w),$$

where $\Psi^X(\cdot)$ denotes the objective function (risk)
Equation:

$$\Psi^X(w) \triangleq -\ln(f_X(w)) + c_2\| y - \Phi w \|^2;$$

our ideal risk would be $\Psi^X(x_{MAP})$.

Instead of performing continuous-valued MAP estimation, we optimize for the MAP in the discretized domain $\widehat{\alpha}^N$. We begin with a technical condition on the input.

Condition 1 [link] We require that the probability density has bounded derivatives
Equation:

$$\left|\frac{d}{dx_n}\ln(f_X(x))\right| < \rho,$$

where $\frac{d}{dx_n}$ is the derivative w.r.t. entry n of x, $n \in \{1, ..., N\}$, and $\rho > 0$.

Let $\widetilde{x}_{MAP}$ be the quantization bin in $\widehat{\alpha}^N$ nearest to x_{MAP}. Condition 1 ensures that a small perturbation from x_{MAP} to $\widetilde{x}_{MAP}$ does not change $\ln(f_X(\cdot))$ by much. We use this fact to prove that $\Psi^X(\widetilde{x}_{MAP})$ is sufficiently close to $\Psi^X(x_{MAP})$ asymptotically.

Theorem 9 [link] Let $\Phi \in \mathbb{R}^{M\times N}$ be an i.i.d. Gaussian measurement matrix where each entry has mean zero and variance $\frac{1}{M}$. Suppose that Condition 1 holds and the aspect ratio $\delta = \frac{m}{n} > 0$, and let the noise $z \in \mathbb{R}^M$ be i.i.d. zero-mean Gaussian. Then for all $\epsilon > 0$, the quantized estimator $\widetilde{x}_{MAP}$ satisfies
Equation:

$$\Psi^X(x_{MAP}) \le \Psi^X(\widetilde{x}_{MAP}) < \Psi^X(x_{MAP}) + N\epsilon$$

almost surely as $N \to \infty$.

Given that Theorem 9 shows that the risk penalty due to quantization vanishes asymptotically in n , we now describe a Kolmogorov-inspired estimator for CS over a quantized grid. We define the energy $\epsilon(w^n)$:
Equation:

$$\epsilon(w^n) \triangleq nH_k(w^n) + c_4\| y^m - \Phi w^n \|^2,$$

where $c_4 = c_2 \log_2(e)$.

Ideally, our goal is to compute the globally minimum energy solution
Equation:

$$x_{MAP}^{H_k} \triangleq \underset{w^n \in \widehat{\alpha}^n}{\operatorname{argmin}} \epsilon\left(w^n\right).$$

Theorem 10 [link] Let X be a bounded stationary ergodic source. Then the outcome w_r^n of Algorithm 1 after r iterations obeys
Equation:

$$\lim_{r \to \infty} \epsilon\left(w_r^n\right) = \min_{v^n \in \widehat{\alpha}^n} \epsilon\left(v^n\right) = \epsilon\left(x_{MAP}^{H_k}\right).$$

We define a stochastic transition matrix $P_{(r)}$ from $\widehat{\alpha}^n$ to itself given by the Boltzmann distribution for super-iteration r. Similarly, $\pi_{(r)}$ defines the stable state distribution on $\widehat{\alpha}^n$ for $P_{(r)}$, satisfying $\pi_{(r)} P_{(r)} = \pi_{(r)}$.

Definition 4 [link]*The Dobrushin's ergodic coefficient* of a Markov chain transition matrix P is denoted by $\delta(P)$ and defined as
Equation:

$$\delta\left(P\right) \triangleq \max_{1 \leq i,j \leq N} \frac{1}{2} \| p_i - p_j \|_1,$$

where p_i denotes the i^{th} row of P, $1 \leq n \leq N$.

From the definition, $0 \leq \delta(P) \leq 1$. Moreover, the ergodic coefficient can be rewritten as
Equation:

$$\delta\left(P\right) = 1 - \min_{1 \leq i,j \leq N} \sum_{k=1}^{N} \min\left(p_{ik}, p_{jk}\right),$$

where p_{ij} denotes the entry of P at the i^{th} row and j^{th} column.

We group the product of transition matrices across super-iterations as
Equation:

$$P_{(r_1 \to r_2)} = \prod_{r=r_1}^{r_2} P_{(r)}.$$

There are two common characterizations for the stable-state behavior of a non-homogeneous MC.

Definition 5 [link] A non-homogeneous MC is called *weakly ergodic* if for any distributions μ and ν over the state space $\mathscr{S}$, and any $r_1 \in \mathbb{N}$,
Equation:

$$\text{limsup}_{r_2 \to \infty} \| \mu P_{(r_1 \to r_2)} - \nu P_{(r_1 \to r_2)} \|_1 = 0.$$

Similarly, a non-homogeneous MC is called *strongly ergodic* if there exists a distribution π over the state space $\mathscr{S}$ such that for any distribution μ over $\mathscr{S}$, and any $r_1 \in \mathbb{N}$,
Equation:

$$\text{limsup}_{r_2 \to \infty} \| \mu P_{(r_1 \to r_2)} - \pi \|_1 = 0.$$

We will use the following two theorems from [link] in our proof.

Theorem 11 [link] A Markov chain is weakly ergodic if and only if there exists a sequence of integers $0 \le r_1 \le r_2 \le \ldots$ such that
Equation:

$$\sum_{i=1}^{\infty} \left(1 - \delta\left(P_{(r_i \to r_{i+1})}\right)\right) = \infty.$$

Theorem 12 [link] Let a Markov chain be weakly ergodic. Assume that there exists a sequence of probability distributions $\left\{\pi_{(r)}\right\}_{i=1}^{\infty}$ on the state space $\mathscr{S}$ such that $\pi_{(r)} P_{(r)} = \pi_{(r)}$. Then the Markov chain is strongly ergodic if
Equation:

$$\sum_{r=1}^{\infty} \| \pi_{(r)} - \pi_{(r+1)} \|_1 < \infty.$$

The rest of proof is structured as follows. First, we show that the sequence of stable state distributions for the Markov chains (MC) used by the MCMC algorithm converges to a uniform distribution over the set of sequences that minimize the energy function as the iteration count t increases. Then, we show using Theorem 11 and Theorem 12 that the non-homogeneous MC used in the MCMC algorithm is strongly ergodic, which by the definition of strong ergodicity implies that MCMC always converges to the stable distribution found above. This implies that the outcome of the MCMC algorithm converges to a minimum-energy solution as $t \to \infty$, completing the proof of Theorem Theorem 10.

We therefore begin by finding the stable state distribution for the non-homogeneous MC used by the MCMC algorithm. At each super-iteration r, the distribution defined as
Equation:

$$\begin{aligned}\pi_{(r)}\left(w^n\right) &\triangleq \frac{\exp\left(-\frac{1}{t_r}\epsilon\left(w^n\right)\right)}{\sum_{z^n\in\widehat{\alpha}^n}\exp\left(-\frac{1}{t_r}\epsilon\left(z^n\right)\right)} \\ &= \frac{1}{\sum_{z^n\in\widehat{\alpha}^n}\exp\left(-\frac{1}{t_r}\left(\epsilon\left(z^n\right)-\epsilon\left(w^n\right)\right)\right)}.\end{aligned}$$

satisfies $\pi_{(r)}P_{(r)}=\pi_{(r)}$. We can show that the distribution $\pi_{(r)}$ converges to a uniform distribution over the set of sequences that minimize the energy function, i.e.,
Equation:

$$\lim_{r\to\infty}\pi_{(r)}\left(w^n\right)=\begin{cases}0 & w^n\notin\mathscr{H},\\ \frac{1}{|\mathscr{H}|} & w^n\in\mathscr{H},\end{cases}$$

where $\mathscr{H}=\left\{w^n\in\widehat{\alpha}^n\ \text{s. t.}\ \epsilon\left(w^n\right)=\min_{z^n\in\widehat{\alpha}^n}\epsilon\left(z^n\right)\right\}$. To show [link], we will show that $\pi_{(r)}\left(w^n\right)$ is increasing for $w^n\in\mathscr{H}$ and eventually decreasing for $w^n\in\mathscr{H}^C$. Since for $w^n\in\mathscr{H}$ and $z^n\in\widehat{\alpha}^n$, $\epsilon\left(z^n\right)-\epsilon\left(w^n\right)\geq 0$, and so for $r_1<r_2$ we have
Equation:

$$\sum_{z^n\in\widehat{\alpha}^n}\exp\left(-\frac{1}{t_{r_1}}\left(\epsilon\left(z^n\right)-\epsilon\left(w^n\right)\right)\right)\geq\sum_{w^n\in\widehat{\alpha}^n}\exp\left(-\frac{1}{t_{r_2}}\left(\epsilon\left(z^n\right)-\epsilon\left(w^n\right)\right)\right),$$

which together with [link] implies $\pi_{(r_1)}\left(w^n\right)\leq\pi_{(r_2)}\left(w^n\right)$. On the other hand, if $w^n\notin\mathscr{H}$, then
Equation:

$$\begin{aligned}&\frac{1}{\pi_{(r)}\left(w^n\right)}\\ =&\sum_{z^n:\epsilon(z^n)\geq\epsilon(w^n)}\exp\left\{-\frac{1}{t_r}\left(\epsilon\left(z^n\right)-\epsilon\left(w^n\right)\right)\right\}\\ +&\sum_{z^n:\epsilon(z^n)<\epsilon(w^n)}\exp\left\{-\frac{1}{t_r}\left(\epsilon\left(z^n\right)-\epsilon\left(y^n\right)\right)\right\}.\end{aligned}$$

For sufficiently small t, the right hand side (more precisely, the second and third lines) of [link] is dominated by the second term (third line), which increases when t decreases, and therefore $\pi_{(r)}\left(w^n\right)$ decreases for $w^n\in\mathscr{H}$ as t increases. Finally, since all sequences $w^n\in\mathscr{H}$ have the same energy $\epsilon(w^n)$, it follows that the distribution is uniform over the symbols in $\mathscr{H}$.

Having shown convergence of the non-homogenous Markov chain's stable state distributions, we now show that the non-homogeneous MC is strongly ergodic. The transition matrix $P_{(r)}$ of the MC at iteration t depends on the temperature t used within MCMC algorithm. We first show that the MC used in the MCMC algorithm is weakly ergodic via Theorem 11; the proof of the following Lemma is given at the end of this section.

Lemma 2 The ergodic coefficient of $P_{(r)}$ for any $r \geq 0$ is upper bounded by
Equation:

$$\delta\left(P_{(r)}\right) \leq 1 - \exp\left\{-\frac{1}{t_r} n \Delta_k\right\},$$

where Δ_k is defined in [link].

Let w_1^n, w_2^n be two arbitrary sequences in $\widehat{\alpha}^n$. The probability of transitioning from a given state to a neighboring state in an iteration within iteration r' of super iteration r of the MCMC algorithm is given by [link], and can be rewritten as
Equation:

$$\begin{aligned} P_{(r,r')}\left(w_{r'} = b \middle| w^{n\backslash r'}\right) &= \Pr\left(w_{r'} = b \middle| w^{n\backslash r'}\right) \\ &= \frac{\exp\left\{-\frac{1}{t_r}\epsilon\left(w_{r'} = b, w^{n\backslash r'}\right)\right\}}{\sum_{b' \in \widehat{\alpha}} \exp\left(-\frac{1}{t_r}\epsilon\left(w_{r'} = b', w^{n\backslash r'}\right)\right)} \\ &= \frac{\exp\left(-\frac{1}{t}\left(\epsilon\left(w_1^{r'-1} b w_{r'+1}^n\right) - \epsilon_{\min,r'}\left(w_1^{r'-1}, w_{r'+1}^n\right)\right)\right)}{\sum_{b' \in \widehat{\alpha}} \exp\left(-\frac{1}{t}\left(\epsilon\left(w_1^{r'-1} b' w_{r'+1}^n\right) - \epsilon_{\min,r'}\left(w_1^{r'-1}, w_{r'+1}^n\right)\right)\right)} \\ &\geq \frac{\exp\left(-\frac{1}{t}\Delta_k\right)}{\left|\widehat{\alpha}\right|}, \end{aligned}$$

where $\epsilon_{\min,r'}\left(w_1^{r'-1} b w_{r'+1}^n\right) = \min_{b' \in \widehat{\alpha}} \epsilon\left(w_1^{r'-1} b' w_{r'+1}^n\right)$. Therefore the smallest probability of transition from w_1^n to w_2^n within super-iteration r is bounded by
Equation:

$$\min_{w_1^n, w_2^n \in \widehat{\alpha}^n} P_{(r)}\left(w_2^n | w_1^n\right) \geq \prod_{r'=1}^{n} \frac{\exp\left(-\frac{1}{t_r}\Delta_k\right)}{\left|\widehat{\alpha}\right|} = \frac{\exp\left(-\frac{1}{t_r} n \Delta_k\right)}{\left|\widehat{\alpha}\right|^n}.$$

Using the alternative definition of the ergodic coefficient given in [link],
Equation:

$$\begin{aligned}\delta\left(P_{(r)}\right) &= 1-\min_{w_1^n,w_2^n\in\widehat{\alpha}^n}\sum_{z^n\in\widehat{\alpha}^n}\min\left(P_{(r)}\left(z^n|w_1^n\right),P_{(r)}\left(z^n|w_2^n\right)\right)\\ &\leq 1-\left|\widehat{\alpha}\right|^n\frac{\exp\left(-\frac{1}{t_r}n\Delta_k\right)}{\left|\widehat{\alpha}\right|^n}=1-\exp\left(-\frac{1}{t_r}n\Delta_k\right).\end{aligned}$$

Using Lemma 2, we can evaluate the sum given in Theorem 11 as
Equation:

$$\begin{aligned}\sum_{r=1}^{\infty}\left(1-\delta\left(P_{(r)}\right)\right) &\geq \sum_{r=1}^{\infty}\exp\left(-\frac{1}{t_r}n\Delta_k\right)\\ &= \sum_{r=1}^{\infty}\frac{1}{r^{1/c}}\\ &= \infty,\end{aligned}$$

and therefore the non-homogeneous MC defined by $\left\{P_{(r)}\right\}_{r=1}^{\infty}$ is weakly ergodic. Now we can use Theorem 12 to show that the MC is strongly ergodic by proving that
Equation:

$$\sum_{r=1}^{\infty}\parallel\pi_{(r)}-\pi_{(r+1)}\parallel_1<\infty.$$

Since we know from earlier in the proof that $\pi_{(r)}\left(w^n\right)$ is increasing for $w^n\in\mathscr{H}$ and eventually decreasing for $w^n\in\mathscr{H}^C$, there exists a $r_0\in\mathbb{N}$ such that for any $r_1>r_0$,
Equation:

$$\begin{aligned}\sum_{r=r_0}^{r_1}\parallel\pi_{(r)}-\pi_{(r+1)}\parallel_1 &= \sum_{w^n\in\mathscr{H}}\sum_{r=r_0}^{r_1}\left(\pi_{(r+1)}\left(w^n\right)-\pi_{(r)}\left(w^n\right)\right)\\ &+ \sum_{w^n\notin\mathscr{H}}\sum_{r=r_0}^{r_1}\left(\pi_{(r)}\left(w^n\right)-\pi_{(r+1)}\left(w^n\right)\right)\\ &= \sum_{w^n\in\mathscr{H}}\left(\pi_{(r_1+1)}\left(w^n\right)-\pi_{(r_0)}\left(w^n\right)\right)\\ &+ \sum_{w^n\notin\mathscr{H}}\left(\pi_{(r_0)}\left(w^n\right)-\pi_{(r_1+1)}\left(w^n\right)\right)\\ &= \parallel\pi_{(r_1+1)}-\pi_{(r_0)}\parallel_1\\ &\leq \parallel\pi_{(r_1+1)}\parallel_1+\parallel\pi_{(r_0)}\parallel_1\\ &= 2.\end{aligned}$$

Since the right hand side does not depend on r_1, then we have that
Equation:

$$\sum_{r=1}^{\infty} \| \pi_{(r)} - \pi_{(r+1)} \|_1 < \infty.$$

This implies that the non-homogeneous MC used by MCMC algorithm is strongly ergodic, and thus completes the proof of Theorem 10.

Compression of nonparametric sources

Consider a stationary ergodic source, where we no longer assume that it is parametric. We need to define notions of probability that fit the universal framework. For this, we study Kac's lemma [link].

We have a stationary source, $\{\ldots, X_{-n}, X_{-n+1}, \ldots, X_0, X_1, \ldots\}$,
$z = X^0_{-\ell+1} = \{X_{-\ell+1}, X_{-\ell+2}, \ldots, X_0\}$. Define N_r as the number of shifts forward of a window of length ℓ until we see z again; this is called recurrence time. Define
Equation:

$$Y_k = \begin{cases} 1 & X^k_{k-\ell+1} = z \\ 0 & \text{else} \end{cases},$$

e.g., $Y_0 = 1$, then N_r is the smallest positive number for which $Y_k = 1$. Note that $\{Y_k\}^{+\infty}_{-\infty}$ is a binary stationary ergodic source. Define $Q_k = \Pr\{Y_k = 1; 1 \le j \le k-1, Y_j = 0 | Y_0 = 1\}$. Then the average recurrence time can be computed, $\mu = \sum_{\ell=1}^{\infty} \ell Q_\ell = E[N_r]$. We can now present Kac's lemma.

Lemma 3 [link]$\mu = \frac{1}{\Pr(Y=1)}$ and $E\left[N_r = \mu | X^0_{-\ell+1} = z\right] = \frac{1}{\Pr(z)}$.

Let $A = \{Y_n = 1 \text{ for some } -\infty < n < +\infty\}$. Because z just appeared, then its probability is positive. We will prove that $\mu = \frac{\Pr(A)}{\Pr(Y_0=1)}$.

Define $B^+ = \{Y_n = 1, \text{for some } 0 \le n < \infty\}$, and $B^- = \{Y_n = 1, \text{for some } n < 0\}$. Then $A = B^+ \bigcup B^- = (B^+ \bigcap B^-) \bigcup \left(B^+ \bigcap (B^-)^C\right) \bigcup \left((B^+)^C \bigcap B^-\right)$. We claim that $\Pr\left(B^+ \bigcap (B^-)^C\right) = \Pr\left((B^+)^C \bigcap B^-\right) = 0$. This can be shown formally, but is easily seen by realizing that if z appears at any time n (say, positive) then it must appear at some negative time n with probability 1, because its probability is positive.

Therefore, we have
Equation:

$$\begin{aligned}\Pr(A) &= \Pr\left(B^{+}\bigcap B^{-}\right)\\ &= \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Pr(Y_j=1, Y_{-k}=1, Y_n=0, -k<n<j)\\ &= \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Pr(Y_{-k}=1)\Pr(Y_j=1, Y_n=0, -k<n<j|Y_{-k}=1)\\ &= \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Pr(Y_{-k}=1)Q_{j+k}\\ &= \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Pr(Y_0=1)Q_{j+k}\\ &= \Pr(Y_0=1)\sum_{i=1}^{\infty}iQ_i\\ &= \Pr(Y_0=1)\mu.\end{aligned}$$

Therefore, $\mu = \frac{\Pr(A)}{\Pr(Y_0=1)}$. We conclude the proof by noting that $\Pr(A)=0$.

Let us now develop a universal coding technique for stationary sources. Recall $H_\ell = \frac{1}{\ell}E\left[-\log\left(\Pr\left(X_1^\ell\right)\right)\right]$. The asymptotic equipartition property (AEP) of information theory [link] gives

Equation:

$$\Pr\left(X_1^\ell : \left|-\frac{1}{\ell}\log\left(\Pr\left(X_1^\ell\right)\right) - H_\ell\right| > \delta\right) < \epsilon(\delta,\ell),$$

where $\lim_{\ell\to\infty}\epsilon(\delta,\ell)=0$. Define in this context a typical set $T(\delta,\ell)$ that satisfies

Equation:

$$2^{-(H_\ell+\delta)\ell} \le \Pr\left(X_1^\ell\right) \le 2^{-(H_\ell-\delta)\ell}.$$

For a typical sequence z, $E\left[N_r\middle|X_1^\ell = z\right] \le 2^{\ell(H_\ell+\delta)}$. Then

Equation:

$$\begin{aligned}\Pr\left(\frac{\log(N_r)}{\ell} \ge H_\ell + 2\delta\right) &= \Pr(z\in T(\delta,\ell))\Pr\left(\frac{\log(N_r)}{\ell} \ge H_\ell + 2\delta | z\in T(\delta,\ell)\right)\\ &+ \Pr(z\notin T(\delta,\ell))\Pr\left(\frac{\log(N_r)}{\ell} \ge H_\ell + 2\delta | z\notin T(\delta,\ell)\right)\\ &\le 2^{-\delta\ell} + \epsilon(\delta,\ell) \xrightarrow{\text{AEP}} 0.\end{aligned}$$

Consider our situation, we have a source with memory of length n and want to transmit X_1^ℓ.

1. Choose $n = 2^{\ell(H_\ell + 2\delta)}$.
2. For $z = X_1^\ell$, find the value of N_r if it appears in memory.
3. If it appears, then transmit a 0 flag bit followed by the value of N_r.
4. Else transmit a 1 followed by the uncompressed z.

Transmitting z via N_r requires $\lceil \log(N_r) \rceil$ bits, and so the expected coding length is
Equation:

$$\Pr\left(\frac{\log(N_r)}{\ell} \leq H_\ell + 2\delta\right)\left(1 + 1 + \log(N_r) + 2^{-\delta\ell}(1 + \ell\log\alpha)\right) + \epsilon(\ell,\delta)(1 + 1 + \ell\log\alpha)$$
$$\leq (2 + \ell(H_\ell + \delta)) + 2^{-\delta\ell}(1 + \ell\log\alpha) + \epsilon(\ell,\delta)(2 + \ell\log\alpha).$$

After we normalize by ℓ, the per symbol length converges to $\frac{2}{\ell} + H_\ell + 2\delta$.

Note that this analysis assumes that the entropy H_ℓ for a block of ℓ symbols is known. If not, then we can have several sets that are each adjusted for some different entropy level.

Universal Coding of the Integers

Coding of an index also appears in other information theoretic problems such as indexing a coset in distributed source coding and a codeword in lossy coding. What are good ways to encode the index?

If the index n lies in the range $n \in \{1, \ldots, N\}$ and N is known, then we can encode the index using $\lceil \log(N) \rceil$ bits. However, sometimes the index can be unbounded, or there could be an (unknown) distribution that biases us strongly toward smaller indices. Therefore, we are interested in a universal encoding of the integers such that each n is encoded using roughly $\log(n)$ bits.

To do so, we outline an approach by Elias [link]. Let n be a natural number, and let $b(n)$be the binary form for n. For example, $b(0) = 0, b(1) = 1, b(2) = 10, b(3) = 11$, and so on. Another challenge is that the length of this representation, $|b(n)|$, is unknown. Let $u(k) = 0^{k-1}1$, for example $u(4) = 0001$. We can now define $e(n) = u(|b(|b(n)|)|)b(|b(n)|)b(n)$. For example, for n=5, we have $b(n) = 101$. Therefore, $|b(n)| = 3, b(|b(n)|) = 11, |b(|b(n)|)| = 2, u(|b(|b(n)|)|) = 01$, and $e(n) = 0111101$. We note in passing that the first 2 bits of $e(n)$ describe the length of $b(|b(n)|)$, which is 2, the middle 2 bits describe $b(|b(n)|)$, which is 3, and the last 3 bits describe $b(n)$, giving the actual number 5. It is easily seen that
Equation:

$$\begin{aligned} |e(n)| &= 2|b(|b(n)|)| + |b(n)| \\ &\leq 2\log\lceil \log(n+1) + 1 \rceil + log\lceil n+1 \rceil. \end{aligned}$$

Sliding Window Universal Coding

Having described an efficient way to encode an index, in particular a large one, we can employ this technique in sliding window coding as developed by Wyner and Ziv [link].

Take a fixed window of length n_w and a source that generates characters. Define the history as $x_1^{n_w}$, this is the portion of the data x that we have already processed and thus know. Our goal is to encode

the phrase $y_1 = x_{n_w+1}^{n_w+L_1}$, where L_1 is not fixed. The length L_1 is chosen such that there exists
Equation:

$$y_1 = x_{n_w+1}^{n_w+L_1} = x_{n_w-m}^{n_w-m+L_1}$$

for some $m \in \{0, 1, \ldots, n_w - 1\}$. For example, if $x = abcbababc$ and we have processed the first 5 symbols $abcba$, then $L_1 = 3$ and $m_1 = 1$, where $abcba$ is the history and we begin encoding after that part.

The actual encoding of the phrase y_1 sends $|f(y_1)|$ bits, where
Equation:

$$|f\left(y_1\right)|= \begin{cases} \lceil \log\left(n_w\right)\rceil + |e\left(L_1\right)|, \log\left(n_2\right) < L_1 \log\left(\alpha\right) \\ \lceil L_1 \log\left(\alpha\right)\rceil + |e\left(L_1\right)|, else \end{cases}.$$

First we encode L_1; then either the index m into history is encoded or y_1 is encoded uncompressed; finally, the first L_1 characters $x_1^{L_1}$ are removed from the history database, and y_1 is appended to its end. The latter process gives this algorithm a sliding window flavor.

This rather simple algorithm turns out to be asymptotically optimal in the limit of large n_w. That is, it achieves the entropy rate. To see this, let us compute the compression factor. Consider x_1^N where $N >> n_w$, $\overline{R}\left(N\right) = \frac{E\left(l\left(x_1^N\right)\right)}{N}$, $N = \sum_{i=1}^{C} |y_i|$, and $|y_i|= L_1$, where C is the number of phrases required to encode x_1^N. The number of bits $l\left(x_1^N\right)$ needed to encoded x_1^N using the sliding window algorithm is
Equation:

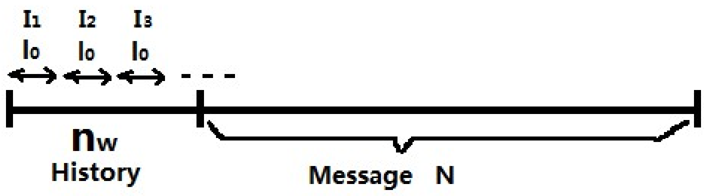
$$l\left(x_1^N\right) = \lceil n_w \log\left(\alpha\right)\rceil + \sum_{i=1}^{C} |e\left(L_i\right)| + 1 + \min\left\{\lceil \log\left(n_w\right)\rceil, \lceil L_i \log\left(\alpha\right)\rceil\right\}.$$

Therefore, the compression factor $\overline{R}\left(N\right)$ satisfies
Equation:

$$\overline{R}\left(N\right) = \frac{\lceil n_w \log\left(\alpha\right)\rceil}{N} + \frac{1}{N}\sum_{i=1}^{C} min\left\{\lceil \log\left(n_w\right)\rceil + 1 + r_1 \log\left(L_i\right), r_2 \log\left(L_i\right)\right\}.$$

Claim 3 **[link]** As $\lim_{n_w \to \infty}$ and $\lim_{N \to \infty}$, the compression factor $\overline{R}\left(N\right)$ converges to the entropy rate H.

A direct proof is complicated, because the location of phrases is a random variable, making a detailed analysis complicated. Let us try to simplify the problem.



Block partitioning in analysis of sliding window algorithm [link].

Take l_0 that divides $n_w + N$ to create $\frac{n_w+N}{l_0}$ intervals, where $l_0 = \frac{\log(n_w)}{H+2\epsilon}$, which is related to the window size in the algorithm from [link]. This partition into blocks appears in [link]. Denote by $w_{l_0}(x) > 0$ the smallest value for which $x_1^{l_0} = x_{-w_{l_0}-l_0+1}^{-w_{l_0}}$. Using Kac's lemma [link],

Equation:

$$\Pr\left(w_{l_0}(x) > n \middle| x_1^{l_0} = z\right) \leq \frac{1}{n \Pr\left(x_1^{l_0} = z\right)}.$$

Claim 4 [link] We have the following convergence as l increases,

Equation:

$$Pr\left(w_{l_0}(x) > 2^{l(H+2\epsilon)}\right) \xrightarrow{l\to\infty} 0.$$

Because of the AEP [link],

Equation:

$$\Pr\left(x_1^l : \left|\frac{-\log\left(Pr\left(x_1^l\right)\right)}{l_0} - H\right| > \epsilon\right) \to 0.$$

Therefore, for a typical input $\Pr\left(x_1^l\right) > 2^{-l_0(H+\epsilon)}$.

Recall that the interval length is $l_0 = \frac{\log(n_w)}{H+2\epsilon}$, and so the probability that an interval cannot be found in the history is

Equation:

$$\frac{2^{-l_0(H+2\epsilon)}}{2^{-l_0(H+\epsilon)}} = 2^{-l_0\epsilon} \to 0.$$

For a long enough interval, this probability goes to zero.

Redundancy of parsing schemes

There are many Ziv-Lempel style parsing algorithms [link], [link], [link], and each of the variants has different details, but the key idea is to find the longest match in a window of length n_w. The length of the match is L, where we remind the reader that $L \approx \frac{\log(n_w)}{H}$.

Now, encoding L requires $\log(n_w)$ bits, and so the per-symbol compression ratio is $\frac{\log(n_w)}{L}$, which in the limit of large n_w approaches the entropy rate H.

However, the encoding of L must also desribe its length, and often the symbol that follows the match. These require length $\log(L) \approx \log(\log(n_w))$, and the normalized (per-symbol) cost is
Equation:

$$\frac{\log(\log(n_w))}{L} = O\left(\frac{\log(\log(n_w))}{\log(n_w)}\right).$$

Therefore, the redundancy of Ziv-Lempel style compression algorithms is proportional to $O\left(\frac{\log(\log(n))}{\log(n)}\right)$, which is much greater than the $O\left(\frac{\log(n)}{n}\right)$ that we have seen for parametric sources. The fundamental reason why the redundancy is greater is that the class of non-parametric sources is much richer. Detailed redundancy analyses appear in a series of papers by Savari (c.f. [link]).

Parsing for Lossy Compression

The parsing schemes that we have seen can also be adapted to lossy compression. Let us describe several approaches along these lines.

Fixed length: The first scheme, due to Gupta et al. [link], constructs a codebook of size $\approx 2^{LR(D)}$ codewords, where L is the length of the phrase being matched and $R(D)$ is the rate distortion function. The algorithm cannot search for perfect matches of the phrase, because this is lossy compression. Instead, it seeks the codeword that matches our input phrase most closely. It turns out that for large L the expected distortion of the lossy match will be approximately D per symbol.

Variable length: Another approach, due to Gioran and Kontoyiannis [link], constructs a single long database string, and searches for the longest match whose distortion w.r.t. the input is approximately D ; the location and length of the approximate match are encoded. Seeing that the database is of length $\approx 2^{LR}$, encoding the location requires $\approx LR$ bits, and the D-match (a match with distortion D w.r.t. the input string) is typically of length $\approx L$, giving a per-symbol rate of $\approx R(D)$ bits.

An advantage of the latter scheme by Gioran and Kontoyiannis [link] is reduced memory use. The database is a string of length $\approx 2^{LR(D)}$, instead of a codebook comprised of $\approx 2^{LR(D)}$ codewords, each of length L. On the other hand, the Gupta et al. algorithm [link] has better RD performance, because it does not need to spend $\approx \log(L)$ bits per phrase to describe its length. An improved algorithm, dubbed the hybrid algorithm by Gioran and Kontoyiannis, constructs a single database and performs fixed length coding for the best match of length L in the database. Therefore, it combines the memory usage of a single database approach with the RD performance of fixed length coding.

Notation

- x - input sequence
- n - length of x
- $x = x_1 x_2 ... x_n$, $x_i^j = x_i x_{i+1} ... x_j$
- α - discrete alphabet
- r - cardinality of α
- $n_x\left(a\right)$ - the number of times that $a \in \alpha$ appears in x
- $P_x\left(a\right)$ - empirical probability
- Q - the true i.i.d. distribution of x
- H - entropy
- $D(\cdot \parallel \cdot)$ - divergence
- $\lceil \cdot \rceil$ - rounding up
- R - rate of source code
- $T_Q\left(\delta\right)$ - set of inputs that are δ-typical with respect to (w.r.t.) Q
- $\mathscr{C}$ - a code
- $\left(\cdot\right)^C$ - complement of set
- $\cdot^T$ - transpose of vector or matrix
- $l(x)$ - length of code in bits
- $x' = S_x$ - step $\forall n \in \mathbb{Z}$, $x'_n = x_{n+1}$
- θ - parameters of parametric source (can contain multiple scalar parameters)
- M - number of states of unifilar source
- $S = \{1, ..., M\}$ - states of unifilar source
- $s_1, ..., s_n,$ - state sequence
- $q(s|s')$ - transition probability
- $S_t = g\left(S_{t-1}, X_{t-1}\right)$ - next state function
- Λ - class of parametric models
- c_n - collection of lossless codes for length-n inputs
- $I(\cdot;\cdot)$ - mutual information
- $r_n\left(l, \theta\right)$ - how far a coding length function l is from the entropy
- $h_2\left(\cdot\right)$ - binary entropy
- $I(\cdot)$ - Fisher information
- $J(\cdot)$ - Jeffreys' prior
- θ_{ML} - maximum likelihood parameter
- R_n^-, R_n^+ - min-max and max-min redundancy

- T - set of leaves of a context tree source
- s - state of context tree
- $n_x\,(s, a)$ - number of times that $a \in \alpha$ appears in context s in x
- T^* - optimal context tree source
- D - maximal depth of tree source
- T_s^* - optimal tree structure to encode symbols whose context ends with s
- $\mathrm{MDL}(s)$ - minimum description length required for encoding these symbols
- $\mathrm{KT}(\cdot, \cdot)$ - Krichevsky-Trofimov coding length
- (y, i) - BWT output consisting the permuted sequence and index
- x - approximate version of x
- α - reproduction alphabet
- $d(\cdot, \cdot)$ - distortion metric
- $d \quad x, x \quad$ - distortion between sequences
- D - distortion level
- $R(D)$ - minimal rate such that x can be described
- t - temperature in simulated annealing
- Z_t - normalization factor in Boltzmann pmf
- N_r - recurrence time